
175 PTAS
69
mi computer
CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR
Editorial Delta, S.A.

# MI COMPUTER CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR


Publicado por Editorial Delta, S.A., Barcelona

Volumen VI-Fascículo 69

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones, Francisco Martín
Asesor técnico: Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Paseo de Gracia, 88, 5.°, 08008 Barcelona
Tels. (93) 215 10 32 / (93) 215 10 50 - Télex 97848 EDLTE


MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.



ISBN: 84-85822-83-8 (fascículo) 84-7598-034-7 (tomo 6)
84-85822-82-X (obra completa)
Depósito Legal: B. 52/1984

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 088505
Impreso en España-Printed in Spain-Abril 1985


Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.


Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.


Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

**Servicio de suscripciones y atrasados** (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 19 425 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 429 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S.A. (Paseo de Gracia, 88, 5.°, 08008 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

Para cualquier aclaración, telefonear al (93) 215 75 21.

**No se efectúan envíos contra reembolso.**



# Un cambio continental

GRAN BRETAÑA 510

PAÍSES ESCANDINAVOS 180

REPÚBLICA FEDERAL DE ALEMANIA 390

FRANCIA 304

HOLANDA 75

BÉLGICA 54

SUIZA 41

ITALIA 160

ESPAÑA 57

UNA UNIDAD = 1 000 ORDENADORES PERSONALES INSTALADOS. ESTADÍSTICA CORTESÍA DE INTELLIGENT ELECTRONICS (EUROPE)

Kevin Jones

## En esta ocasión analizaremos los pros y los contras que implica la traducción de programas del inglés a otras lenguas

Hasta hace poco tiempo casi todos los programas de software estaban escritos en inglés. El inglés era el idioma estándar de la informática debido al predominio de Estados Unidos en este campo durante la década de los cincuenta y los sesenta.

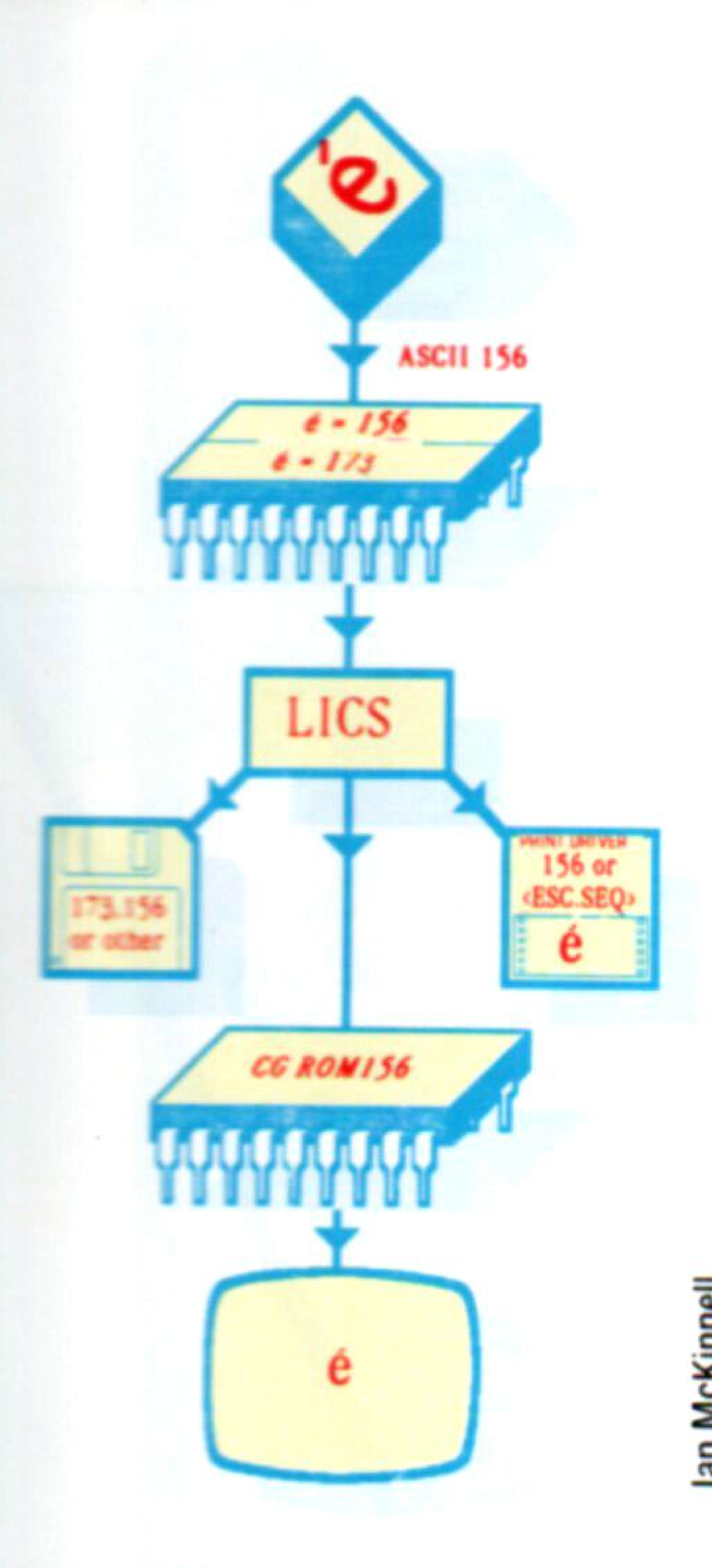
Sin embargo, la introducción del microordenador, a finales de los años setenta y principios de los ochenta, provocó grandes problemas en los países de habla no inglesa. Ello representó un gran obstáculo para la difusión de la microinformática a través de la Europa continental y creó un círculo vicioso. Las empresas no invertían en software que no comprendían ni podían utilizar con facilidad, y las firmas de software no gastaban el dinero necesario para traducir sus programas al idioma del país cuando las ventas previsibles no eran suficientes para cubrir los costos de desarrollo. El resultado fue que Gran Bretaña, disfrutando de la enorme ventaja de compartir un idioma común con Estados Unidos, se convirtió en el mercado más desarrollado de toda Europa para los microordenadores.

En la actualidad la situación está empezando a cambiar. Las firmas de software han comprendido el enorme mercado potencial de Europa continental y han empezado a traducir sus programas a los idiomas de los países en los cuales esperan vender sus productos. Lotus Software, como una de las mayores proveedoras de programas de gestión para IBM, fue una de las primeras empresas que produjo traducciones a otras lenguas.

El *Lotus 1-2-3* y el *Symphony* figuran entre los paquetes más vendidos de la nueva generación de "software integrado". Estos paquetes por lo general incorporan una hoja electrónica, base de datos y algunas capacidades para gráficos y tratamiento de textos. Los datos se pueden pasar entre cada una de estas aplicaciones, permitiendo, por ejemplo, almacenar información en una base de datos para tratarla en una hoja electrónica y poder luego incorporarla a un documento.

La traducción de uno de tales programas del inglés a, por ejemplo, el italiano, podría parecer relativamente sencilla: todas las instrucciones que aparezcan en la pantalla se han de traducir al idioma apropiado. Sin embargo, de inmediato surgen diversas dificultades. En primer lugar, si el texto está incorporado en el propio código fuente, encontrar el texto en 120 K de código en el que tanto el texto

**Ordenadores personales en Europa**
Debido a que Gran Bretaña partió con la ventaja de compartir una lengua común con Estados Unidos, hasta el momento las empresas británicas han superado ampliamente a las de Europa continental desde el punto de vista de la instalación de ordenadores personales. Parece ser, sin embargo, que el notable crecimiento de las ventas en Gran Bretaña está terminando justo cuando se produce el despegue del mercado en los otros países europeos; se calcula que para 1988 la República Federal de Alemania habrá superado a Gran Bretaña en la cantidad total de máquinas instaladas




Ian McKinnell

**ASCII internacional**
Para tratar con los juegos de caracteres específicos que emplean los distintos idiomas, Lotus ha desarrollado el LICS (Lotus International Character Set: juego internacional de caracteres Lotus), en el cual hay un código para cada carácter no perteneciente al idioma inglés. En un teclado francés, por ejemplo, la pulsación de la letra *é* (como en *café*) generaría un código 156; éste corresponde a 173 en LICS. Cuando este carácter se ha de enviar a la pantalla, el LICSA lo vuelve a decodificar a 156; la impresora podría comprender el código 156, o bien necesitar que se le enviara una secuencia de control tal como <e>, <ESC>, <BSPACE>, <'>. El texto se puede guardar en códigos ASCII ingleses, en LICS o en códigos ASCII de otro país

como el programa están representados por números constituye una tarea muy complicada. En segundo lugar, si el texto traducido es más extenso que el texto original en inglés (lo que sucede casi siempre) el programa necesitará más bytes para almacenarlo. Esto alterará las direcciones de todo el código subsiguiente, convirtiendo, por lo tanto, en un sin sentido los bucles y las llamadas a subrutinas.

Otro problema es la sintaxis. Cuando un usuario inglés desea operar sobre un archivo, la sintaxis es COMMAND (instrucción) seguido de FILENAME (nombre del archivo). Sin embargo, este enfoque no es el normal en otros idiomas. En alemán, por ejemplo, lo lógico es entrar primero el nombre del archivo, seguido por la instrucción. Se plantea un problema similar en la forma de entrar las fechas, lo que ha sido fuente de problemas incluso en Gran Bretaña. Tanto el *1-2-3* como el *Symphony* permiten el empleo de fechas en fórmulas para calcular cambios en los valores a través del tiempo. En Estados Unidos, el método normal para entrar los datos es mes/día/año. Sin embargo, en Gran Bretaña y en muchas otras partes de Europa el formato estándar para fechas es día/mes/año. A menos que el paquete de software se pueda manipular de modo que tenga en cuenta las diferencias en cuanto a la forma de entrar instrucciones y datos, el resultado será, en el mejor de los casos, confuso o, en el peor, un sin sentido total.

Los editores de software deben asimismo tener en cuenta los diferentes juegos de caracteres. El alfabeto francés incluye letras tales como *é* y *à*, mientras que el alemán y las lenguas escandinavas incluyen la letra *ä*. Para complicar aún más las cosas, los diferentes alfabetos colocan estas letras en otro orden, haciendo estragos en cualquier rutina incapaz de resolver estas diferencias.

# Diseño de programa

Al traducir el *Symphony* a las principales lenguas europeas, Lotus decidió que la única forma razonable de abordar el problema era diseñar el programa de modo que permitiera una fácil traducción. Este enfoque no se adoptó con el paquete *1-2-3* anterior y, a consecuencia de ello, la empresa tuvo grandes dificultades para traducirlo. No obstante, el *Symphony* se ha traducido con total éxito al francés, al alemán y a las lenguas escandinavas y la empresa está trabajando ahora en una versión italiana.

Para superar los problemas de localización del texto en el código y tratar de comprimir las nuevas palabras en el espacio disponible, Lotus ha adoptado una construcción modular del programa. Dentro de éste hay dos divisiones: el código fuente que contiene las rutinas del programa y un segmento de datos que incluye el área de textos. Este sistema de aislar el texto del código fuente se conoce como *localización*. La organización del programa de acuerdo a este formato resuelve dos de las dificultades principales. Primero, el hecho de tener el texto en un segmento separado significa que se puede reservar espacio extra para cualquier diferencia en cuanto a extensión del texto, y éste puede ser extraído del programa con mucha más facilidad. Una utilidad extrae del código las áreas de texto; éstas se pueden entonces traducir y volver a incluir en el segmento de datos. Como ventaja adicional, el texto se puede reacomodar dentro de la sección de datos para dar cabida a las diferencias de sintaxis que requiera cada idioma.

Al traducir el texto propiamente dicho hay que considerar todavía otro punto. El *Symphony* permite entrar las instrucciones simplemente pulsando el primer carácter del nombre de la instrucción. Por lo tanto, cada instrucción debe empezar con una letra diferente. Además, las limitaciones de espacio significan que en un paquete en el cual la traducción literal sea demasiado larga, quizá sea necesario hacer alguna concesión.

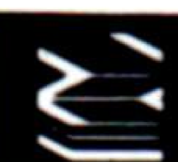
También pueden surgir problemas cuando se traduce entre juegos nacionales de caracteres. Ya hemos visto que los diversos países poseen letras diferentes en sus alfabetos. Lo que agrava el problema es que para los códigos no existe ninguna estandarización acordada con carácter internacional. En los días de la cinta de papel, cuando eran comunes los códigos de siete bits, el estándar ASCII se utilizaba en todo el mundo casi sin ninguna excepción. Con el advenimiento del microordenador y de los códigos de ocho bits, la estandarización desapareció debido a que cada fabricante produjo su propia versión del "estándar" ASCII.

Esta práctica la han adoptado distintos países. Al adaptar el teclado del ordenador a su propio juego de caracteres, muchas naciones han sustituido algunos caracteres ingleses por sus propias letras. Por consiguiente, la comunicación entre ordenadores configurados para idiomas diferentes se está convirtiendo en un enorme problema. El código ASCII en alemán podría significar algo totalmente diferente en castellano.

Esta confusión la agravaron aún más los traductores del *Symphony*, ya que muchas de las instrucciones de una única pulsación de tecla utilizadas en el programa eran caracteres tales como @, que no existe en los teclados no ingleses. La solución que encontró para este problema la propia IBM en su PC es mantener pulsada la tecla ALT y digitar el código ASCII decimal en el teclado numérico, ¡anulando, de este modo, la ventaja que reportaba el tener instrucciones para las cuales sólo se había de pulsar una tecla!

Lotus decidió que la única forma de abordar este problema era desarrollar su propio juego de códigos, que se denomina LICS (Lotus International Character Set: juego internacional de caracteres Lotus). Este juego, de 250 caracteres, contiene todas las letras utilizadas en la mayoría de las lenguas europeas y está almacenado en todas las copias del *Symphony*. La traducción supone configurar el programa de modo que el código que reciba un teclado de lengua no inglesa se traduzca a LICS, y el programa pueda, en consecuencia, comprenderlo. Para simplificar el proceso de imprimir caracteres que no aparezcan en el teclado, Lotus ha conseguidor reducir estos caracteres a una única pulsación de la tecla ALT y un número entre 0 y 9.

El proceso de traducción de un paquete de gestión es una operación costosa y que consume mucho tiempo. La traducción lleva un promedio de nueve meses y es extraordinariamente cara. No obstante, las firmas de software ya no pueden permitirse el lujo de ignorar un mercado de 300 millones de personas como es el de Europa continental. A pesar de la inversión que requiere la traducción de un paquete de software, los beneficios que ésta reporta bien valen el esfuerzo.



# Desde el Más Allá

**Nos corresponde abordar el resto de la rutina del túnel y diseñar una subrutina para que aparezcan fantasmas al azar en el "bosque encantado" de nuestro juego**

En el capítulo anterior analizamos los escenarios especiales que poseen una entrada al túnel: en estos escenarios se le ofrece al jugador la oportunidad de penetrar en el túnel o de retroceder hasta el sendero que lo condujo a la entrada. Si el jugador elige entrar en el túnel, entonces se llama a una nueva subrutina, en la línea 4655. Veamos ahora la subrutina que maneja la opción en la cual el jugador penetra en el túnel. Esta subrutina está escrita de acuerdo a ciertas reglas que estableció el diseñador del juego. En primer lugar, el jugador sólo puede atravesar el túnel si lleva consigo el farol; además, debe encender el farol para alumbrar su camino.

Dado que el jugador ha de poder impartir instrucciones mientras permanezca en el interior del túnel, la subrutina debe empezar con una secuencia que acepte la entrada de una instrucción y la descomponga para su proceso. Podemos permitir que el jugador utilice algunas de las instrucciones de entrada normales, como RECOGER, DEJAR, LISTA o FIN, pero en este punto hemos de tener cuidado. Por cuanto concierne al puntero del escenario, P, el jugador se halla todavía en la boca del túnel y, por lo tanto, está en condiciones de AVANZAR en ciertas direcciones permitidas. Por consiguiente, mientras nos encontremos dentro del túnel, debemos suprimir las instrucciones AVANZAR.

Al retornar de la subrutina "instrucciones normales", de haberse generado una instrucción AVANZAR se establecerá la "bandera de movimiento" (MF) y el valor de P habrá cambiado. Este efecto se puede anular simplemente restaurando a P el valor que tenía antes de que se llamara a la subrutina "instrucciones normales".

Habiendo manipulado satisfactoriamente las instrucciones normales, podemos pasar a considerar las instrucciones especializadas necesarias para esta situación en particular. Se puede permitir que el jugador utilice una instrucción RETROCEDER para regresar hasta la entrada al túnel que traspasó al entrar. La única otra instrucción que permitiremos es ENCENDER, o una variación, USAR. Si la instrucción impartida no es ninguna de éstas, la rutina producirá un mensaje general NO COMPRENDO antes de desandar el bucle en espera de otra instrucción.

Si la instrucción es ENCENDER o USAR, entonces tendremos que efectuar varias comprobaciones antes de obedecer la instrucción:

1. ¿Es el objeto especificado un objeto válido?
2. ¿Lleva consigo el jugador el objeto especificado?
3. ¿Es el farol el objeto especificado?

Si la respuesta a todas estas preguntas es "sí", al jugador se le permitirá atravesar el túnel hasta el otro extremo, puesto que se han satisfecho las condiciones para atravesarlo. Estas comprobaciones del objeto ya nos resultan familiares. De hecho, son casi idénticas a las empleadas en las rutinas RECOGER y DEJAR. Por lo tanto, para llevar a cabo estas verificaciones podemos utilizar rutinas escritas anteriormente.

```
4700 REM ** ENTRAR AL TUNEL **
4705 SN$="ENTRAS EN EL TUNEL PERO ESTA DEMASIADO OSCURO
     COMO"
4710 SN$=SN$+"PARA ENCONTRAR EL CAMINO":GOSUB5500
4725 PRINT:INPUT"INSTRUCCIONES";IS$
4730 GOSUB2500:REM DESCOMPONER INSTRUCCION
4732 :
4735 IF F=0 THEN 4725:REM INSTRUCCION NO VALIDA
4740 OP=P:GOSUB3000:REM INSTRUCCIONES NORMALES
4745 IF MF=1 THEN SN$="ESTA TAN OSCURO QUE SOLO PUEDES
     VER":P=OP
4747 IF MF=1THENSN$=SN$+" LA ENTRADA DEL
     TUNEL":GOSUB5500:MF=0:GOTO 4725
4750 IF VF=1 THEN 4725:REM INSTRUCCION OBEDECIDA
4755 IF VB$="RETROCEDER" AND P=4 THEN MF=1:P=6:
     RETURN
4760 IF VB$="RETROCEDER" AND P=1 THEN MF=1:P=9:
     RETURN
4762 IF VB$<>"USAR" AND VB$<> "ENCENDER"THEN SN$="NO
     COMPRENDO"
4765 IF VB$<>"USAR" AND VB$<> "ENCENDER"THEN
     GOSUB5500:GOTO4725
4777 :
4780 REM ** BUSCAR FAROL **
4790 GOSUB5300:REM OBJETO VALIDO?
4795 OV=F:GOSUB5450:REM LLEVA EL OBJETO?
4797 IF F=0 THEN SN$="NO HAY NINGUN"+W$:GOSUB5500:
     GOTO4725
4800 IF HF=0 THEN SN$="TU NO TIENES EL "+IV$(F,1):GOSUB5500:
     GOTO4725
4810 REM ** EL OBJETO ES EL FAROL? **
4815 IF F<>2 THEN SN$="LA "+IV$(F,1)+" NO
     SIRVE":GOSUB5500:GOTO4725
4835 REM ** EXITO **
4840 SN$="USAS EL FAROL PARA ILUMINARTE EL CAMINO A TRAVES
     DEL TUNEL"
4845 SN$=SN$+" Y FINALMENTE SALES POR LA SALIDA.":
     GOSUB5500
4850 IF P=1 THEN MF=1:P=4:RETURN
4855 IF P=4 THEN MF=1:P=1:RETURN
```

## Fenómenos sobrenaturales

Además de tener escenarios especiales, como las entradas al túnel, podemos programar peligros o acontecimientos al azar. Llegados a este punto, en el desarrollo de nuestro juego *El bosque encantado* no hemos mencionado fantasmas, ni los mismos aparecen en el mapa del mundo de aventuras para el juego. En cambio, los fantasmas se le aparecen al azar al jugador a medida que va recorriendo el bosque y sólo se los puede mantener a raya emprendiendo una acción extravagante. Antes de analizar detalladamente la rutina "de los fantasmas",

consideremos cómo podemos incorporar en la estructura del programa principal las rutinas para generar apariciones aleatorias. El bucle principal del programa llama a una subrutina de la línea 2700 para constatar si un nuevo escenario es especial en algún sentido o no. Éste es también el mejor lugar para incorporar el siguiente trozo de código, que tiene como finalidad decidir si el programa debe generar fantasmas al azar:

```
2707 REM ** FANTASMA AL AZAR **
2710 IF P >4 AND RND(1)<0.1 THEN GOSUB 4290:RETURN
```

La línea 2710 asegura, en primer lugar, que el escenario actual no haya sido ya designado como especial, dado que si los fantasmas aparecieran en la mitad de rutinas especiales harían que la vida resultara muy complicada. Si el escenario es común, utilizando la instrucción RND existe una posibilidad entre 10 de que el programa produzca un fantasma. Las instrucciones RND generan números *seudoaleatorios*, así llamados porque el patrón de números generados desde la conexión a la red del ordenador es predecible. Para hacer que la secuencia sea menos predecible, utilizamos la instrucción RND con un operando negativo en el caso del BBC Micro y del Commodore 64, y la instrucción RANDOMISE para el Spectrum (véase "Complementos al BASIC" en la página contigua).

```
207 R=RND(-1)
```

Si se llama a la rutina de los fantasmas, entonces entramos en otro escenario especial en el cual el jugador se enfrenta a la aparición fantasmal. La rutina sigue el procedimiento usual: genera un mensaje inicial, solicita una instrucción y la descompone en el verbo y el resto de la frase. Las instrucciones normales son tratadas por la subrutina estándar; pero nuevamente se suprime la instrucción AVANZAR: un mensaje informa al jugador que, al haberse quedado paralizado por el terror, no se puede mover.

## Red de seguridad

En esta etapa se pueden tratar nuevas instrucciones. Al igual que las otras rutinas de tratamiento de escenarios especiales, la calidad del juego terminado depende de cuánto esfuerzo de programación se dedique al diseño de estas rutinas. Cualquier instrucción que no sea directamente útil en la rutina se puede tratar mediante la red de seguridad NO COMPRENDO. No obstante, con un esfuerzo de programación adicional, podemos manejar instrucciones que cabría esperar del jugador pero que no serán de ayuda en su situación. En la rutina "fantasmas" se emplea un ejemplo de este enfoque.

Si un jugador se encuentra con un fantasma, su primera idea podría ser LUCHAR o MATAR (¡en el caso de que uno pudiera matar a los fantasmas!). La rutina "fantasmas" trata estas dos instrucciones llamando a una subrutina especial. Esta subrutina simplemente visualiza un mensaje que señala que estas instrucciones no le son de ayuda al jugador, pero lo hace de tal forma que es sustancialmente más atractiva que limitarse a expresar un escueto NO COMPRENDO.

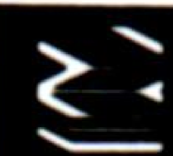
```
4290 REM **** S/R FANTASMA AL AZAR ****
4295 SF=1:GC=0
4300 SN$="SIENTES QUE UN ESCALOFRIO TE RECORRE"
4305 SN$=SN$+" LA COLUMNA VERTEBRAL. DE PRONTO UNA
     APARICION BLANCA"
4310 SN$=SN$+" SALE DE ATRAS DE LOS ARBOLES Y"
4315 SN$=SN$+" AVANZA HACIA TI":GOSUB5500:REM
     FORMATO
4320 :
4325 SN$="EL FANTASMA SE ACERCA MAS Y MAS":GOSUB
     5500
4330 GC=GC+1:IF GC>4 THEN GOSUB4455:REM
4335 PRINT:INPUT"INSTRUCCIONES";IS$
4340 GOSUB2500:REM DESCOMPONER INSTRUCCION
4345 IF F=0 THEN 4325:REM SIGUIENTE INSTRUCCION
4350 OP=P:GOSUB3000:REM ANALIZAR INSTRUCCION
4355 IF MF=1 AND VB$="AVANZAR" THEN GOSUB4400: GOTO 4325
4357 IF MF=1 AND VB$="MIRAR" THEN GOSUB2000:GOSUB2300:
     GOTO4325
4360 IF VF=1 THEN 4325:REM SIGUIENTE INSTRUCCION
4365 REM ** PALABRAS DE INSTRUCCION NUEVAS **
4370 IF VB$="MATAR" OR VB$="LUCHAR" THEN GOSUB4425:GOTO
     4325
4375 :
4385 IF VB$="CANTAR" THEN GOSUB4500:RETURN
4390 SN$="NO COMPRENDO":GOSUB5500:GOTO4325
4395 :
4400 REM ** TRATAR DE MOVERSE **
4405 SN$="ESTAS PARALIZADO POR EL TERROR Y NO PUEDES"
4410 SN$=SN$+" MOVERTE...TODAVIA":MF=0:GOSUB5500: P=OP
4415 RETURN
4420 :
4425 REM ** LUCHAR O MATAR **
4430 SN$="EL FANTASMA ES UN SER SOBRENATURAL"
4435 SN$=SN$+" Y SE RIE DE TUS DEBILES INTENTOS"
4440 SN$=SN$+" POR HERIRLO":GOSUB5500
4445 RETURN
4450 :
4455 REM ** MUERTE **
4460 SN$="EL DOLOR QUE SIENTES EN EL PECHO SE VUELVE
     INSOPORTABLE"
4465 SN$=SN$+" Y TE DESPLOMAS SOBRE EL SUELO CUBIERTO DE
     HOJAS DEL BOSQUE.":GOSUB5500
4470 SN$="TU ESPIRITU SE DESPRENDE DE TU CUERPO INERTE"
4475 SN$=SN$+" Y TE ALEJAS FLOTANDO ENTRE LA NIEBLA PARA
     UNIRTE"
4480 SN$=SN$+" A LAS OTRAS ALMAS ATORMENTADAS DE"
4485 SN$=SN$+" EL BOSQUE ENCANTADO.":GOSUB5500
4490 END
```

## Una pequeña trampa

De impartirse alguna de las instrucciones normales, o instrucciones que no le sirven de nada al jugador, la rutina las obedecerá, si le es posible, y saltará hacia atrás del bucle para una nueva instrucción. Esta rutina tiene una pequeña trampa, porque lleva la cuenta del número de instrucciones impartidas por el jugador mientras se mide con el fantasma. De impartir más de cuatro instrucciones, el fantasma mata al jugador. El único medio de que dispone el jugador para escapar es CANTAR una canción. Si el jugador opta por cantar, se le pide que elija entre tres canciones, una de las cuales (escogida al azar) apaciguará al fantasma. Sin embargo, si el jugador elige una canción errónea, su espíritu se unirá al ejército de almas atormentadas que se han extraviado en *El bosque encantado*:

```
4500 REM ** CANTAR **
4505 SN$="SABES TRES CANCIONES. CUAL ELEGIRIAS ?":
     GOSUB5500
4510 SN$="1) EL TEMA MUSICAL DE 'LOS
     CAZAFANTASMAS'":GOSUB5500
4515 SN$="2) 'HAY UN FANTASMA EN MI CASA'":GOSUB5500
4520 SN$="3) 'BAJANDO POR EL RIO SWANEE'":GOSUB5500
4525 PRINT:INPUT"ELIGE UNA";C$
4530 IF VAL(C$)>3 OR VAL(C$)<1 THEN PRINT:PRINT"NO
     VALE":GOTO4525
4535 CR=INT(RND(1)*3)+1
4537 IF CR<>VAL(C$)THEN GOSUB4542:REM CANCION EQUIVOCADA
4540 GOSUB4565:REM CORRECTA
4542 REM **** S/R CANCION EQUIVOCADA ****
4545 SN$="EL FANTASMA TIENE UNA ESPECIAL AVERSION POR "
4550 SN$=SN$+"ESA MELODIA Y SE ABALANZA SOBRE TI."
     GOSUB5500
4555 GOSUB 4455:REM MUERTE
4560 :
4565 REM ** CANCION CORRECTA **
4570 SN$="EL FANTASMA SE APACIGUA AL OIRTE CANTAR LA
     MELODIA"
4575 SN$=SN$+" Y SE EVAPORA EN EL AIRE":GOSUB5500
4580 RETURN
```

Fantasmas realizados por Liz Dixon



# Listados Digitaya

```
2690 IF P=37 THEN2780:REM TABLA VECTORES
2700 IF P>7 THEN 2750:REM BICHO AZAR

2740 REM ** BICHO AL AZAR **
2750 RA=RND(TI)
2760 IF RA<0.05THEN GOSUB 5420:REM BICHO
2770 RETURN
2780 REM ** TABLA VECTORES **
2790 SF=1
2800 SN$="ERES LLEVADO A GRAN VELOCIDAD HASTA UN NUEVO
     ESCENARIO":GOSUB5880
2810 FOR J=1TO1000:NEXT:REM PAUSA
2820 P=INT(RND(TI)*40+7)
2830 MF=1:RETURN

4550 REM **** ALU ****
4560 SF=1
4570 RN=INT(RND(TI)*3+1)
4580 IF RN=1 THEN CD$="AND"
4590 IF RN=2 THEN CD$="OR"
4600 IF RN=3 THEN CD$="NOT"
4610 SN$="MONTADOS SOBRE LA PARED HAY TRES BOTONES
     ROTULADOS"
4620 SN$=SN$+"'AND', 'OR' Y 'NOT'. SE PUEDE GANAR ACCESO
     AL"
4630 SN$=SN$+"ACUMULADOR PULSANDO EL BOTON
     ADECUADO"
4640 GOSUB5880:REM FORMATO
4650 :
4660 REM ** INSTRUCCIONES **
4670 PRINT:INPUT"INSTRUCCIONES";IS$
4680 GOSUB1700:GOSUB1900:REM ANALIZAR
4690 IF MF=1 THEN RETURN:REM IRSE
4700 IF VF=1 THEN 4670:REM SIGUIENTE INSTRUCCION
4710 IF VB$="USAR"OR VB$="PULSAR"THEN4740
4720 PRINT" NO COMPRENDO":GOTO4670
4730 :
4740 REM ** INSTRUCCION VALIDA **
4750 IF VB$="PULSAR" THEN 4930
4760 REM ** LA INSTRUCCION ES 'USAR' **
4770 GOSUB5730:REM ES VALIDO EL OBJETO
4780 IF F=0THENPRINT"NO HAY NINGUN ";NN$:GOTO4670:REM
     SIGUIENTE INSTRUCCION
4790 :
4800 REM ** ES EL OBJETO EL LIBRO DE CODIGOS **
4810 IF F=7 THEN4850:REM OK
4820 SN$="TU "+IV$(F,1)+"NO SIRVE DE NADA":GOSUB5880
4830 GOTO4670:REM SIGUIENTE INSTRUCCION
4840 :
4850 OV=7:GOSUB5830:REM LLEVA EL LIBRO DE CODIGOS
     CONSIGO
4860 IF HF=1THEN4900:REM OK LO LLEVA
4870 SN$="TU NO TIENES EL "+IV$(7,1)
4880 GOSUB5880:GOTO4670:REM SIGUIENTE INSTRUCCION
4890 :
4900 SN$="ABRES EL LIBRO DE CODIGOS Y ENCUENTRAS LA
     PALABRA ' "+CD$+" ' ESCRITA DENTRO"
4910 GOSUB5880:GOTO4670:REM SIGUIENTE INSTRUCCION
4920 :
4930 REM ** LA INSTRUCCION ES PULSAR **
4940 IFNN$="AND"OR NN$="OR" OR NN$="NOT"THEN4970
4950 SN$="NO HAY NINGUN "+NN$:GOSUB5880:GOTO4670:REM
     SIGUIENTE INSTRUCCION
4960 :
4970 REM ** CORRECTO O INCORRECTO **
4980 IFNN$=CD$ THEN GOSUB5100:RETURN
4990 GOSUB 5010 :RETURN
5000 :
5010 REM ** S/R INCORRECTO **
5020 SN$="INCORRECTO, SE ABRE UNA TRAMPA Y TE
     ENCUENTRAS DE VUELTA"
5030 SN$=SN$+" EN LA MEMORIA PRINCIPAL"
5040 GOSUB 5880:REM FORMATEAR
5050 IF RN=1 THEN P=39
5060 IF RN=2 THEN P=35
5070 IF RN=3 THEN P=29
5080 MF=1:RETURN
5090 :
5100 REM ** S/R CORRECTO **
5110 SN$="SE ABRE LA PUERTA QUE CONDUCE AL ACUMULADOR"
5120 SN$=SN$+" Y TU LA ATRAVIESAS":GOSUB 5880
5130 P=30:MF=1:RETURN

5420 REM **** BICHO AL AZAR ****
5430 SF=1
5440 SN$="UN BICHO ENORME Y ASQUEROSO SE ASOMA POR
     DETRAS DE UN CHIP"
5450 SN$=SN$+" Y SE ABALANZA SOBRE TI":GOSUB 5880
5460 :
5470 REM ** INSTRUCCIONES **
5480 PRINT:INPUT"INSTRUCCIONES";IS$
5490 GOSUB1700:GOSUB1900:REM ANALIZAR
5500 IF MF=1THENMF=0:PRINT"NO PUEDES
     MOVERTE...TODAVIA":GOTO5480
5510 IF VF=1 THEN5480:REM SIGUIENTE INSTRUCCION
5520 IF VB$="MATAR"ORVB$="LUCHAR"THEN5550
5530 PRINT"NO COMPRENDO":GOTO5480
5540 :
5550 REM ** LA INSTRUCCION ES LUCHAR/MATAR **
5560 RA=RND(TI)
5570 IFRA<0.5 THEN GOSUB5600
5580 GOSUB5670:RETURN
5590 :
5600 REM **** S/R EL BICHO TE MATA ****
5610 SN$="LUCHAS CON EL BICHO. TE ARROJA UNA LLUVIA DE"
5620 SN$=SN$+" DE ERRORES DE PROGRAMA QUE HACEN
     ESTRAGOS EN TU CEREBRO."
5630 SN$=SN$+" FINALMENTE YA NO PUEDES ABSORBER MAS Y
     TE ESTALLA LA CABEZA."
5640 GOSUB5880
5650 END
5660 :
5670 REM **** S/R TU MATAS AL BICHO ****
5680 SN$="LUCHAS CON EL BICHO Y A PESAR DE QUE LA PELEA ES
     DURA"
5690 SN$=SN$+" FINALMENTE LO VENCES Y LOGRAS
     SOBREVIVIR.":GOSUB5880
5700 RETURN
```



# Complementos al BASIC

**Spectrum:**

En ambos programas, reemplace SN$ por S$, IS$ por T$, IV$(,) por V$(,), VB$ por B$, CD$ por C$ y NN$ por R$.

En el listado de *El bosque encantado* sustituya las siguientes líneas:

```
207 RAND
4815 IF F<>2 THEN LET S$="EL":LET
     A$=V$(F,1):GOSUB7000
4816 IF F<>2 THEN LET S$=S$+"NO
     SIRVE DE NADA":GOSUB5500:GOTO 4725
```

En el listado de *Digitaya* reemplace las líneas siguientes:

```
2750 LET RN=RND(1)
2820 LET P=INT(RND(1)*40+7)
4570 LET RN=INT(RND(1)*3+1)
4820 LET S$="TU":LET A$=V$(F,1):
     GOSUB 8500
4825 LET S$=S$+" NO SIRVE DE
     NADA":GOSUB 5880
5560 LET RA=RND(1)
```

**BBC Micro:**

En el listado de *El bosque encantado* reemplace estas líneas:

```
207 RND(-TIME)
4535 CR=RND(3)
```

En el listado de *Digitaya* reemplace estas líneas:

```
2750 RN=RND(1)
2820 P=RND(40)+7
4570 RN=RND(3)
5560 RA=RND(1)
```

# Dibujos cicloides

## Iniciamos una serie en que estudiaremos el uso del LOGO en la creación de patrones geométricos

A lo largo del curso habíamos ofrecido un método sencillo para dibujar un círculo utilizando el LOGO:

```
TO CIRCULO
  REPEAT 360 [FORWARD 1 RIGHT 1]
END
```

Esto dará una aproximación bastante acertada a lo que es un círculo. No obstante, el dibujo se realiza con una extraordinaria lentitud, si bien se lo puede acelerar un poco escondiendo la tortuga. Si en su pantalla este procedimiento no se parece a un círculo, entonces deberá volver a determinar la proporción de tamaños: habrá de experimentar una y otra vez hasta obtener un círculo y no una elipse.

Por supuesto, en realidad CIRCULO no dibuja un círculo. Dibuja un polígono de 360 lados; pero, en la mayoría de los casos, ésta es una aproximación muy aceptable. De hecho, muchas veces un polígono de 60 lados, o incluso de 30, ya es bastante adecuado y resulta mucho más rápido de dibujar. En este proyecto nuestros círculos serán polígonos de 60 o 120 lados, pero usted puede modificar esto si así lo desea. Los números mayores darán mayor detalle; con los números menores se obtendrá un dibujo más rápido.

En primer lugar, vamos a considerar qué es en realidad un *cicloide*. Imagínese un círculo que rueda a lo largo de una línea recta. Marque un punto de la circunferencia de su círculo imaginario y luego rastree el camino que sigue este punto mientras el círculo se desplaza por la línea. El camino resultante es lo que se conoce como *cicloide*. Utilizaremos esta definición como ayuda para construir un programa que nos dibuje uno.

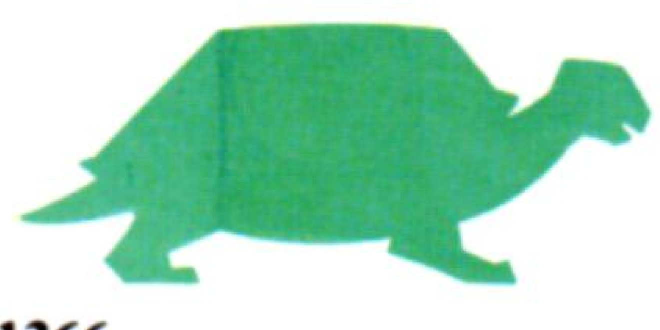
Como una primera aproximación al cicloide, tomaremos "instantáneas" después de cada giro de 6° de un círculo que gire a lo largo de una línea que atraviese la pantalla. Cuando el círculo gira 6°, se mueve (2 × pi × radio ÷ 60) unidades a lo largo de la línea. De modo que la coordenada $x$ del centro del círculo se habrá incrementado en la misma proporción (la coordenada $y$, por supuesto, no se verá afectada). Al mismo tiempo, la orientación de la línea que une el "punto de dibujo" con el centro del círculo se habrá incrementado en 6°.

La estrategia utilizada en el programa implica tres tareas sencillas:

1. mover el centro del círculo;
2. colocar la tortuga en el centro;
3. apuntarla en la dirección correcta;
4. moverla hacia adelante por la longitud del radio.

Con ello la tortuga llega hasta la siguiente posición del punto de dibujo. En este momento dibujamos un punto en la pantalla y después repetimos todo el proceso.

El procedimiento PREPARARPANTALLA es la primera llamada a procedimiento desde CICLOIDE: se encarga de algunos detalles menores necesarios para la visualización. Las principales tareas de PREPARARPANTALLA son determinar la proporción de tamaños (usted necesitará un valor diferente del nuestro) y seleccionar la modalidad NOWRAP (no desplazamiento), de modo que el programa se detenga cuando la curva se salga de la pantalla.



```
TO MOVERCENTRO
  MAKE "XCENT :XCENT + :PASO
END

TO PUNTO
  PD
  FORWARD 1
  BACK 1
  PU
END
```

Si, en lugar de tomar un punto de la circunferencia del círculo generador, rastreamos el camino efectuado por un punto dentro del círculo, entonces obtenemos lo que se conoce como un *cicloide abreviado*. Si tomamos un punto exterior al círculo, pero unido a él, obtenemos otra clase de cicloide: un *cicloide prolongado*. Para observar estos efectos podemos modificar CICLOIDE para que tome una entrada que represente la distancia del punto de dibujo respecto a la circunferencia. Los valores positivos dan cicloides abreviados; los valores negativos, cicloides prolongados:

```
TO CICLOIDE
  PREPARARPANTALLA
  MAKE "ANGULOPASO 6
  MAKE "PI 3.14
  MAKE "RADIO 15
  MAKE "CIRCUNFERENCIA 2* :PI* :RADIO
  MAKE "PASO :CIRCUNFERENCIA/(360/:ANGULO
  PASO)
  MAKE "XCENT(-150)
  CIC 0
END

TO PREPARARPANTALLA
  .ASPECT 0.93
  NOWRAP
  DRAW
  PENUP
  HT
END

TO CIC :ANG
  MOVERCENTRO
  SETXY :XCENT 0
  SETH :ANG
  FORWARD :RADIO
  PUNTO
  CIC :ANG+:ANGULOPASO
END
```



```
TO CICLOIDE :DESPL
  PREPARARPANTALLA
  MAKE "ANGULOPASO 6
  MAKE "PI 3.14
  MAKE "RADIO 15
  MAKE "CIRCUNFERENCIA 2* :PI* :RADIO
  MAKE "PASO :CIRCUNFERENCIA/(360/
  :ANGULOPASO)
  MAKE "XCENT(-150)
  MAKE "DISTANCIA :RADIO-:DESPL
  CIC 0
END

TO CIC :ANG
  MOVERCENTRO
  SETXY :XCENT 0
  SETH :ANG
  FORWARD :DISTANCIA
  PUNTO
  CIC :ANG+:ANGULOPASO
END
```

## Unir los puntos

El marcar los puntos con puntos, como hemos venido haciendo hasta ahora, nos proporciona una forma sencilla de visualizar lo que está sucediendo, pero obtendríamos diagramas más atractivos si pudiéramos unir los puntos entre sí para formar una curva. El procedimiento UNIR dibuja una línea entre dos puntos:

```
TO UNIR :A :B
  ESTPOS :A
  PD
  ESTPOS :B
  PU
END

TO ESTPOS :POS
  SETXY FIRST :POS LAST :POS
END
```

El procedimiento se utiliza con las coordenadas de los dos puntos dados en la llamada. Por ejemplo, una llamada posible es UNIR [12 34][67 89]. En nuestro programa de cicloides, deberemos llevar un registro de la posición antigua del punto, y después unirlo con la posición actual. El resultado final de nuestro programa perfeccionado para dibujar cicloides es:

```
TO CICLOIDE:DESPL
  PREPARARPANTALLA
  MAKE "ANGULOPASO 6
  MAKE "PI 3.14
  MAKE "RADIO 15
  MAKE "CIRCUNFERENCIA 2* :PI*
  :RADIO
  MAKE "PASO :CIRCUNFERENCIA/(360/
  :ANGULOPASO)
  MAKE "XCENT(-150)
  MAKE "DISTANCIA :RADIO -
  :DESPL
  MAKE "POSANT LIST :XCENT
  :DISTANCIA
  CIC 0
END

TO CIC :ANG
  MOVERCENTRO
  SETXY :XCENT 0
  SETH :ANG
  FORWARD :DISTANCIA
  MAKE "POSNUE POS
  UNIR :POSANT :POSNUE
  MAKE "POSANT :POSNUE
  CIC :ANG+ :ANGULOPASO
END

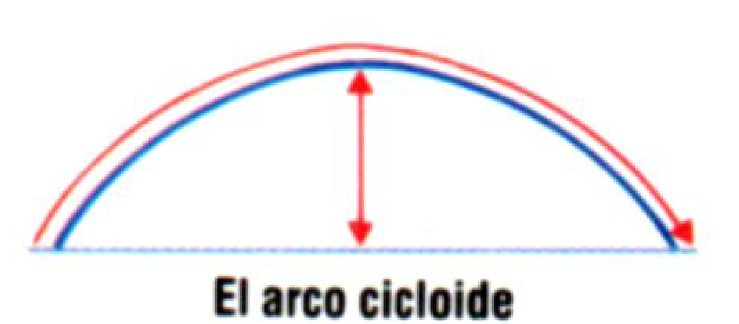
TO POS
  OUTPUT LIST XCOR YCOR
END
```

Quizá le interese realizar algunos experimentos con estos procedimientos. Por ejemplo, los libros de texto de matemáticas afirman que ¡la longitud de un arco de un cicloide es igual al perímetro de un cuadrado circunscrito en el círculo generador! Intente modificar los procedimientos para dibujar cicloides que le permitan comprobar este teorema.

Si posee un LOGO que incorpore sprites, una forma diferente (y mejor) de escribir el programa sería establecer el punto de dibujo como un sprite. Este procedimiento tiene una ventaja: usted siempre podría averiguar la situación del punto mediante el empleo de TELL y después XCOR e YCOR.



El cuadrado circunscrito



El arco cicloide

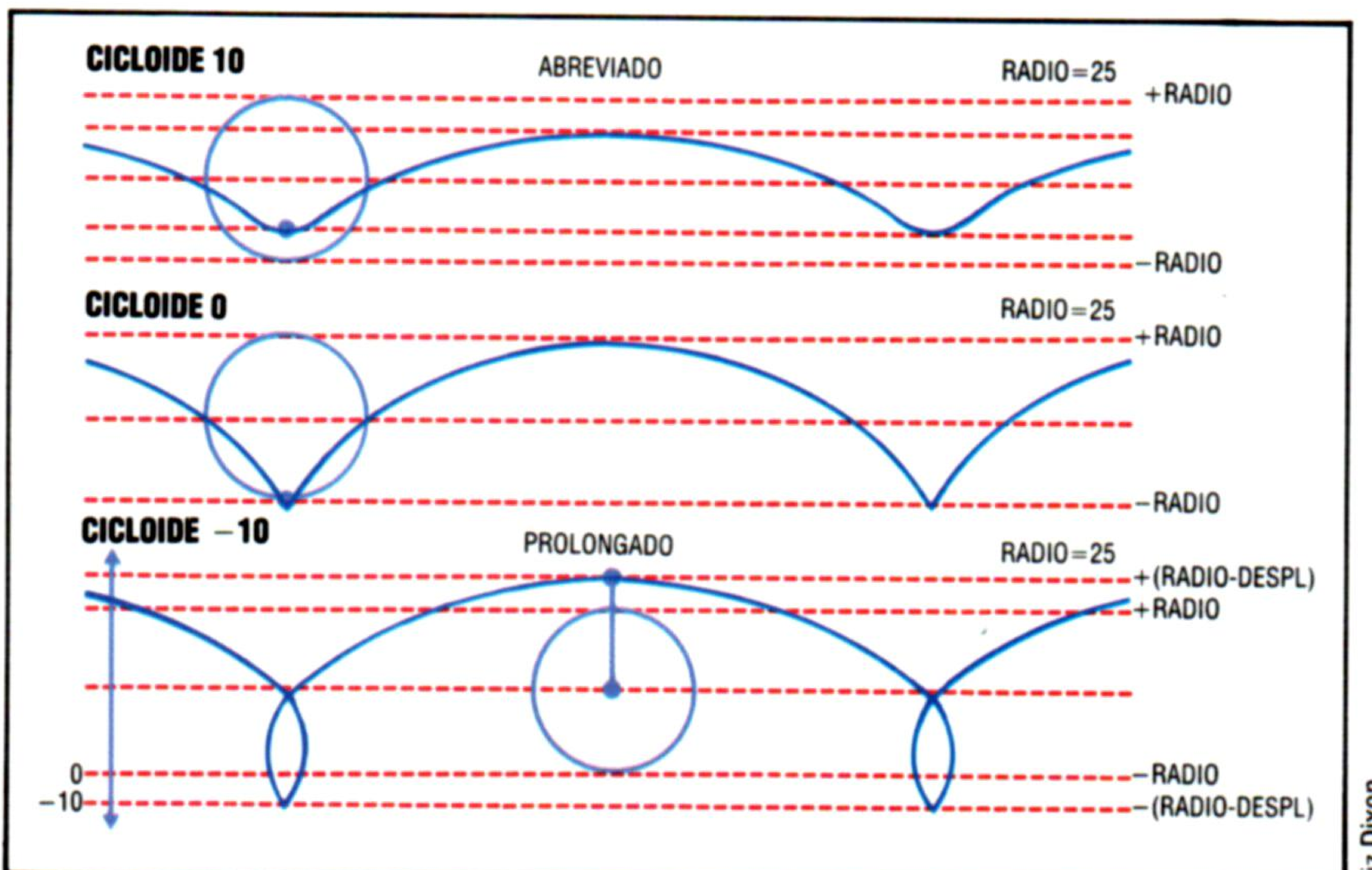
### Complementos al LOGO

Para todas las versiones LCSI:

La sintaxis IF es diferente; por ejemplo: IF :A = 120 [STOP].
SETPOS y POS existen como primitivas.
SETXY deberá sustituirse por SETPOS (que requiere una lista como entrada).
Utilice CS por DRAW.
Para NOWRAP utilice FENCE (FENCE no existe en el LOGO Atari, de manera que emplee WINDOW y luego <BREAK> para detener el procedimiento).

Para establecer la proporción de tamaños utilice:

.SETSCR en el Atari;
SETSCRUNCH en el Apple;
SETSCRUNCH, seguida de una lista, en el Spectrum




**Rodando**
Un cicloide es la curva descrita por el movimiento de un punto en un radio fijo de un círculo que rueda a lo largo de una línea recta. La naturaleza de la curva difiere en función de si el punto se halla dentro, fuera o en el perímetro del círculo

# Generación de hipercicloides

Nuestra investigación de patrones geométricos en LOGO comenzó con la generación de cicloides, las curvas que describen los círculos al rodar sobre una línea recta; sin embargo, con ello no se agotan, de ninguna manera, las posibilidades del círculo móvil.

En lugar de desplazarse a lo largo de una línea recta, el círculo generador podría girar dentro de otro círculo; el camino que recorrería el punto de dibujo en este caso sería un *hipercicloide*.

El problema continúa siendo básicamente el mismo; aún necesitamos mover el centro del círculo generador y luego pasar al punto de dibujo de la circunferencia. Sin embargo, ahora necesitamos llevar el registro de dos ángulos de paso. Uno (ANGULOPASO) es para calcular el camino del centro del círculo generador, y el otro (ORIENTPASO) lleva el registro de la orientación del punto dibujado con respecto al centro de este círculo. Puesto que el centro del círculo más pequeño gira, el punto dibujado gira en la dirección contraria. Se puede demostrar que los tamaños de los dos ángulos de paso están relacionados si aplicamos la fórmula que transcribimos a continuación:

```
ANGULOPASO=ORIENTPASO X
(RADIO1/RADIO2–1)
```

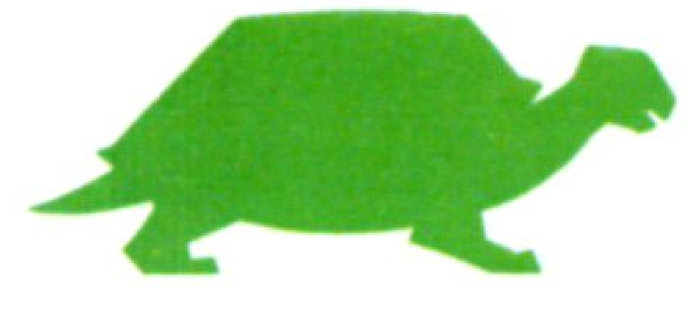
donde RADIO1 es el radio del círculo fijo y RADIO2 es del que gira.

El procedimiento HIPERCICLOIDE toma como entrada RADIO2, permitiéndonos, por tanto, rastrear varios hipercicloides diferentes.

```
TO HIPERCICLOIDE :RADIO2
  PREPARARPANTALLA
  MAKE "PI 3.14
  MAKE "RADIO1 60
  MAKE "DIFERENCIA :RADIO1 –
  :RADIO2
  MAKE "ORIENTPASO 6
  MAKE "CIRCUNFERENCIA 2*:PI*
  :DIFERENCIA
  MAKE "PASO:CIRCUNFERENCIA/(360/
  :ORIENTPASO)
  MAKE "ANGULOPASO :ORIENTPASO*
  (:RADIO1/:RADIO2–1)
  MAKE "CENTRO LISTA 0
  :DIFERENCIA
  MAKE "ORIENT 0
  MAKE "XCENT 0
  MAKE "POSANT LIST :XCENT
  :RADIO1
  HCIC 0
END

TO HCIC :ANG
  MOVERCENTRO2
  ESTPOS POS
  SETH :ANG
  FORWARD :RADIO2
  MAKE "POSNUE POS
  UNIR :POSANT :POSNUE
  MAKE "POSANT :POSNUE
  HCIC :ANG – :ANGULOPASO
END

TO MOVERCENTRO2
  SETXY 0 0
  SETH :ORIENT
  FORWARD :DIFERENCIA
  MAKE "CENTRO POS
  MAKE "ORIENT :ORIENT+
  :ORIENTPASO
END
```



Hay un caso especial interesante: si el radio del círculo que rueda es la mitad del radio del círculo fijo, ¡el hipercicloide se convierte en una línea recta! De este modo, el movimiento dentro de un círculo se transforma en movimiento a lo largo de una línea recta.

Tal vez desee modificar los procedimientos para averiguar lo que sucede si el punto se halla dentro del círculo o fuera del mismo.

# Otro procedimiento: el "cosido de curvas"

El *cosido de curvas* es otra forma de desarrollar algunas formas interesantes a partir de círculos. Tome dos círculos concéntricos y trácelos en una gran cantidad de arcos iguales, supongamos 120. Numere los puntos y después únalos, de uno en uno, a los puntos del otro círculo de acuerdo con alguna regla sencilla; por ejemplo, $x$ "se une a" $2x$. Los resultados pueden ser sorprendentes.

Esta actividad en realidad se puede llevar a cabo con aguja e hilo, por lo cual a menudo se alude a ella como *cosido de curvas*. También se puede realizar con lápiz y papel; pero, obviamente, preferiríamos que usted utilizara el LOGO.

Ésta es nuestra versión para un programa de cosido de curvas:

```
TO PREPARACION
  MAKE "RADIOA 80
  MAKE "RADIOB 60
  DRAW
  HT
  PENUP
  DIBUJARLO 0 0
END

TO DIBUJARLO :A :B
  IF :A = 120 THEN STOP
  UNIR PTA :A PTB :B
  MAKE "A :A+1
  MAKE "B 2* :A
  DIBUJARLO :A :B
END

TO PTA :NUM
  SETXY 0 0
  SETH :NUM*3
  FORWARD :RADIOA
  OUTPUT POS
END

TO PTB :NUM
  SETXY 0 0
  SETH :NUM*3
  FORWARD :RADIOB
  OUTPUT POS
END
```

Es posible que le interese investigar los patrones generados por otras reglas, tales como $x \rightarrow 3x$, $x \rightarrow 4x$, etc.



# Compañero de clases

**El Link 480Z, la más reciente creación de Research Machines, es un ordenador concebido especialmente para la enseñanza**

El Link 480Z se vende en una versión normal, que es la que analizaremos aquí, y en una versión para conectar en red. A primera vista parece muy distinto de su predecesor, el 380Z. Mientras que éste se compone de una gran caja metálica de color negro que contiene el ordenador y las unidades de disco conectados mediante un cable al teclado externo, el Link 480Z posee una sólida carcasa plástica, con el teclado incorporado y las unidades de disco suministradas en una unidad externa opcional. El 480Z no es una máquina normal (mide 520 × por 330 × 80 mm), pero su aspecto estilizado es más agradable que el del funcional y más bien feo 380Z.

La máquina posee un teclado estándar QWERTY tipo máquina de escribir y las teclas son firmes, con una seguridad de tacto que las hace ideales para el tratamiento de textos. Las teclas de control, incluyendo una de salto de línea y Repeat para funciones de edición en pantalla, están situadas a la izquierda y derecha del trazado QWERTY. La única objeción que se le puede hacer al teclado es que la tecla Return quizá sea demasiado pequeña, lo que dificulta su uso.

En el lado derecho del teclado hay un grupo de teclas para control del cursor. En cada una de las esquinas del grupo hay una tecla de función programable; los fines de éstas dependen de la aplicación que se ejecute.

Una gran selección de puertas para interface, situadas en la parte posterior de la máquina, permiten conectar el ordenador a una amplia gama de periféricos. En el extremo izquierdo hay un conector hembra RF, que permite enchufar la máquina a un televisor normal. A la derecha está el botón de RESET.

El Link 480Z posee dos conectores distintos para pantalla: un conector DIN de cinco patillas, que permite conectar el ordenador a la gama de pantallas Microvitec, y, encima del mismo, un conector DIN de ocho patillas para otros tipos de pantallas TTL y RGB. Entre las puertas para pantallas y para cassette hay una interface accesoria. Ésta es una puerta de entrada/salida en serie que permite la conexión de dispositivos externos.

A la derecha de la puerta para cassette se halla la puerta de entrada/salida en paralelo, para la conexión de dispositivos en paralelo (impresoras, p. ej.). Si bien la interface no es una Centronics estándar, es compatible con ésta, lo que significa que si bien están presentes todas las líneas necesarias para un dispositivo Centronics, éstas no están situadas en el orden correcto. Un pequeño reajuste del cableado ofrecería una puerta Centronics totalmente estándar.

El Link 480Z posee asimismo un par de puertas en serie RS232 que permiten conectar la máquina en interface a dispositivos tales como impresoras en serie y las unidades de disco gemelas. Junto a las puertas en serie hay 10 interruptores DIP. El primer interruptor, señalado con una R, permite que el operador desactive el interruptor RESET. Del mismo modo, el segundo interruptor permite activar o desactivar el altavoz interno, situado debajo del teclado.

Los ocho interruptores DIP situados en el extremo izquierdo de este grupo permiten que el usuario establezca la dirección dentro de la red; éstos son leídos como un número binario para dotar al ordenador de una identificación cuando se lo conecta en red. Dado que el 480Z posee ocho de estos interruptores, permite la conexión en red de hasta 256 máquinas diferentes. El cable para la red está instalado en un enchufe hembra de video en la parte posterior del ordenador, lugar donde también hay un ventilador para mantener refrigerado el ordenador, un interruptor de on/off, un fusible y el cable de toma de corriente.

## Las unidades de disco

La unidad de disco gemela MD2 está separada del ordenador. Sorprendentemente, tratándose de un micro moderno, el modelo estándar se conecta a la máquina a través de una interface en serie y no en paralelo, y se enchufa en la segunda puerta RS232. A pesar de esto, la velocidad de transferencia es de 38,5 Kbaudios, comparable a la de muchos micros con transferencia de datos en paralelo. Las unidades gemelas utilizan los discos flexibles estándares de 5 1/4 pulgadas; éstos son de doble cara y doble densidad y están rotulados A, B, C y D. La carcasa

**Aspecto elegante**
El teclado sesgado y la carcasa plástica le confieren a la máquina un aire más elegante que su predecesora, el 380Z, si bien de acuerdo a las pautas modernas sigue siendo una máquina grande. Ello se debe a que dentro de la máquina hay dos niveles de placas de circuito impreso, una para las funciones principales del ordenador y otra para trabajar en red




Chris Stevens

de la unidad es del mismo plástico resistente que la del ordenador, y en funcionamiento es sumamente silenciosa en comparación con otras máquinas de oficina que se venden al doble de precio.

Detrás de las unidades hay un par de conectores RS232, uno para conectar al 480Z y el otro para conectar dispositivos "en margarita"; también poseen su propia fuente de alimentación eléctrica. El sistema de archivo en disco, que administra la transferencia de datos hacia y desde el ordenador, está instalado dentro de la carcasa de la unidad y no dentro de la propia máquina. Este empleo de unidades de disco "inteligentes" significa que el ordenador puede estar cumpliendo otras funciones mientras la gestión de los discos corre a cuenta de las propias unidades, con lo cual la memoria queda para uso del sistema.

Cuando se conecta la máquina, se le solicita al usuario que entre el BASIC ampliado basado en ROM o bien que vea el menú HELP (de ayuda). Pulsando H (de Help) se visualiza la lista de opciones disponibles basadas en ROM. Éstas se refieren fundamentalmente al sistema de entrada/salida. El operador puede optar por cargar programas del sistema ya sea desde cassette o desde disco, o cargar el sistema de red. También se pueden seleccionar la velocidad de la cassette o las opciones de impresora. Hay una opción Front Panel (básicamente, un monitor de memoria) que permite al usuario examinar y modificar los registros del procesador y las posiciones de la memoria. Relacionada con esta opción está la instrucción Jump (saltar), que permite pasar el control a una dirección de la memoria. Por ejemplo, la instrucción J103 le pasa el control al vector de "arranque en caliente".

## Resolución de pantalla

El 480Z dispone de numerosas modalidades de resolución de pantalla. Éstas van desde la pantalla para textos de 80 por 25 hasta la visualización de resolución ultra alta, de 640 por 192 (si bien ésta sólo admite dos colores en pantalla). Hay, además, tres modalidades de color que, en resolución media, soportan la gama completa de 16 colores.

Al igual que el 380Z, el Link 480Z utiliza el microprocesador Z80, que le permite ejecutar una amplia gama de software disponible, incluyendo, por supuesto, el sistema operativo CP/M. La disponibilidad de software quizá haya sido la causa primordial por la cual Research Machines decidiera continuar con este chip en vez de adoptar un procesador más moderno. Si bien la empresa sostiene que el 380Z y el 480Z tienen compatibilidad de software, ciertos programas llamados desde BASIC generan un error de disco cuando el 480Z intenta leer el disco.

En el interior de la máquina se ha dejado espacio para la adición de chips extras. Aunque esta facilidad no es comparable con el 380Z, que está diseñado de modo que las placas extras se puedan instalar con suma facilidad, sí significa que se pueden añadir aplicaciones basadas en ROM, como puede ser el convertidor de digital a analógico para permitir la conexión de la puerta accesoria a un dispositivo analógico.

Con el ordenador viene un disco de sistema que incluye numerosos programas de demostración, una versión de BASIC con instrucciones para gestión de disco y el sistema operativo CP/M.

**RAM adicional**
Dado que el procesador Z80 sólo puede direccionar 64 K de RAM, estos chips no los utiliza directamente el procesador sino que se emplean para acelerar la operación del sistema de red

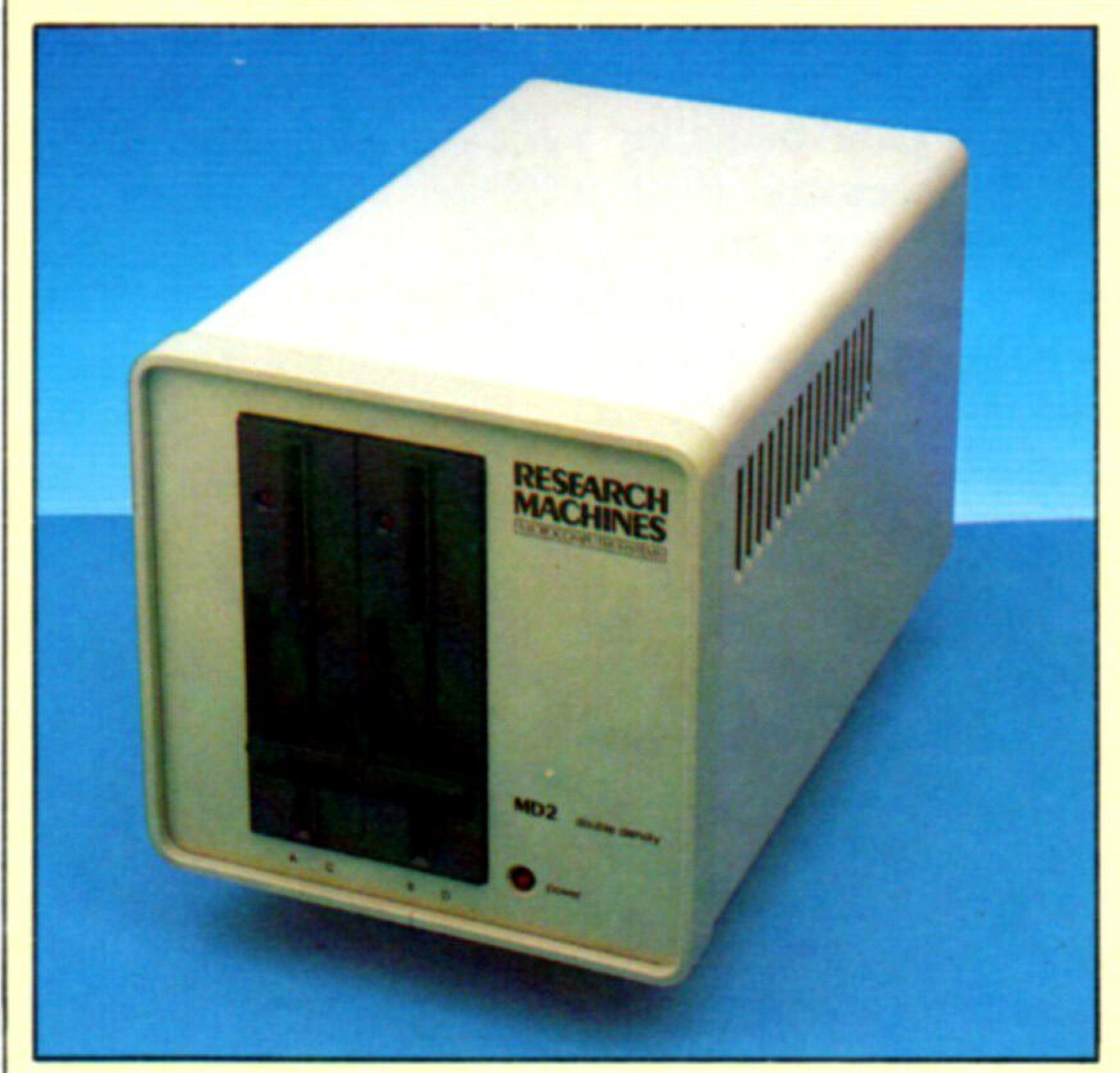
**Fuente de alimentación eléctrica**
El sistema es más grande que el de la mayoría de los micros comparables. La carcasa de metal que la rodea actúa a modo de disipador

**Z80**
El 480Z utiliza como CPU el chip Z80A de 8 bits

**64 K de RAM**
Estos chips contienen la RAM a la que puede acceder el procesador Z80

**Altavoz incorporado**
Se puede desactivar mediante los interruptores DIP situados en la parte posterior de la máquina

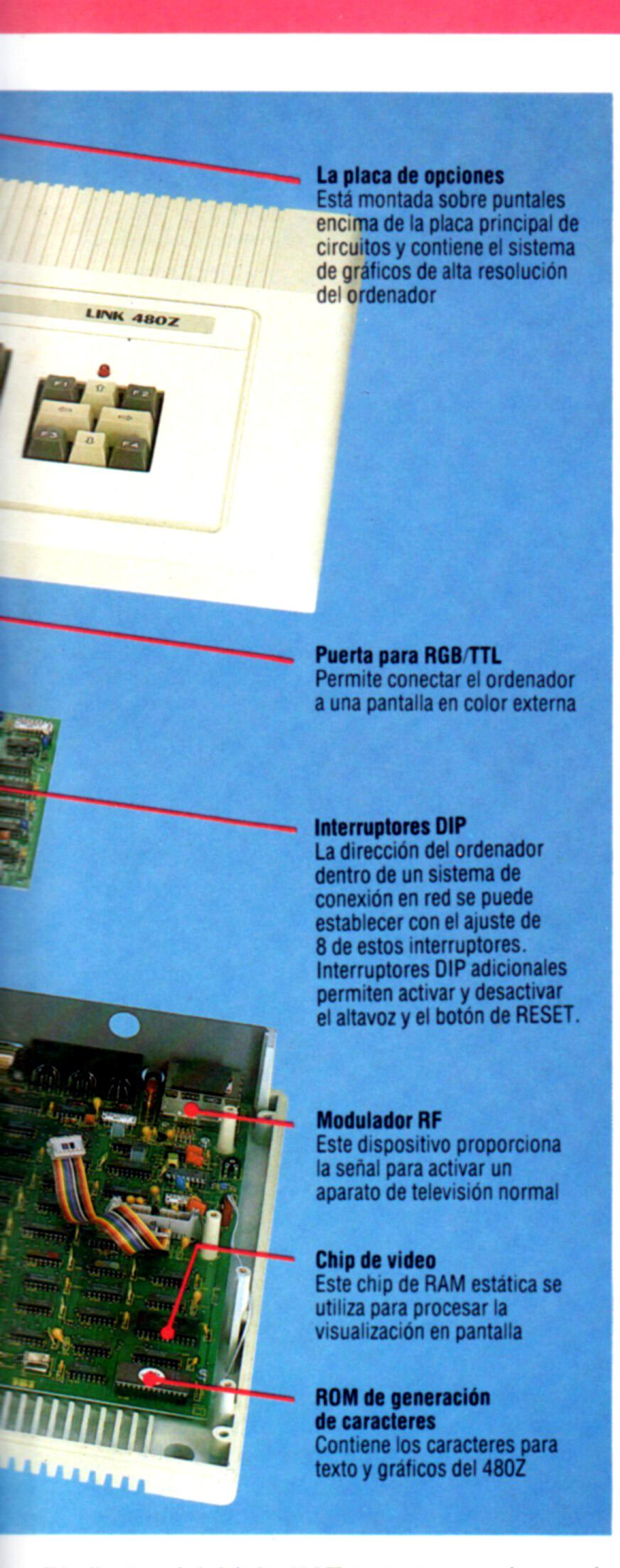
**ROM de BASIC**
Estos chips contienen el BASIC incorporado




## Unidades gemelas

Cada unidad es de doble cara, dando un total de cuatro caras a las cuales acceder. Éstas, siguiendo la convención CP/M, están rotuladas como A, B, C y D. La unidad acepta discos de doble densidad, lo cual le permite a la máquina almacenar el doble de información en cada cara de los discos. Además, Research Machines ha producido una unidad de densidad cuádruple






El diseño del Link 480Z parece sugerir que la intención de Research Machines ha sido desarrollar un ordenador para satisfacer a los usuarios escolares que, no necesitando de la sólida flexibilidad del 380Z, sí requieren una máquina adaptable de propósito general. En este sentido no cabe duda de que la empresa ha acertado. Sin embargo, es una decepción encontrarse con que la máquina introduce tan pocas innovaciones, y su enorme tamaño continúa siendo un misterio. Al mantener el chip Z80, la empresa decidió que la disponibilidad de software a un precio asequible sobrepasa con creces cualquier otra ventaja que hubiera podido obtener si escogía un procesador más moderno de 16 bits. Por otra parte, los lotes de software de los paquetes escolares son una ganga. No obstante, en una época en la que el IBM-PC se está convirtiendo en el estándar para las máquinas de oficina, la elección del Z80, en lugar del chip Intel 8088, parece traslucir una cierta estrechez de miras.

## Paquete escolar

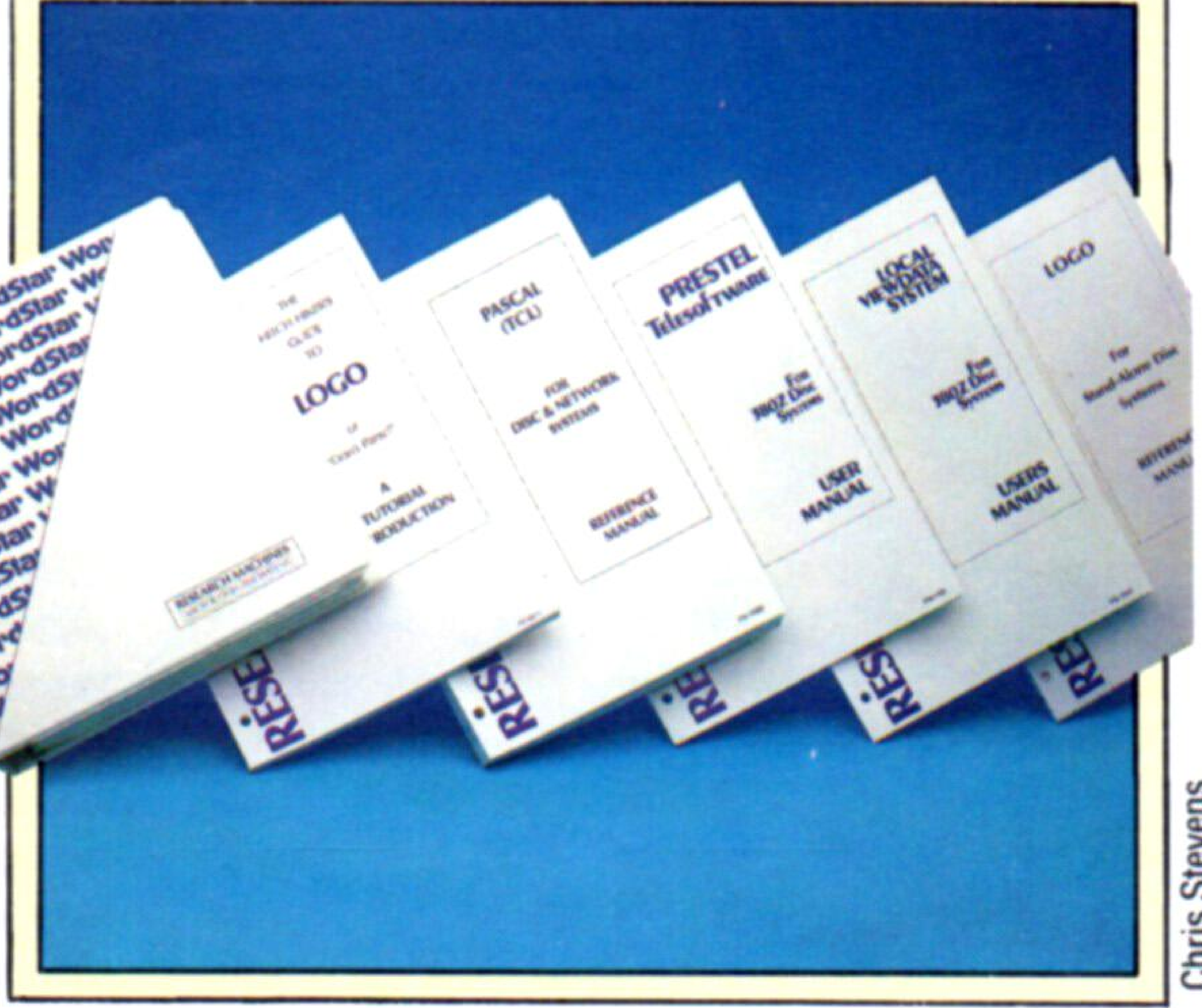
El lote de software del paquete escolar que se entrega con el 480Z ofrece unas prestaciones excelentes. Se suministran 12 paquetes basados en disco, todos de gran valor educativo.

En el lote se incluyen cuatro lenguajes. El SBAS es una versión de BASIC estructurado y está considerado como una implementación excelente. Contiene una amplia gama de estructuras de control para el flujo del programa, incluyendo WHILE...ENDWHILE, CASE... ENDCASE e IF...ENDIF. Dispone asimismo de procedimientos y de variables globales y locales. La máquina también soporta una amplia implementación del PASCAL, que le ofrecerá al estudiante un conocimiento operativo completo de este lenguaje. La documentación que se proporciona con el lenguaje no es de fácil lectura, pero es muy detallada. También se ofrece LOGO en una versión excelente, si bien algunas instrucciones no son estándares. Por ejemplo, esta versión utiliza la instrucción BUILD (construir) en lugar de TO para crear procedimientos. Existe también una implementación parcial del LOGO, denominada ARROW. Para la programación de bajo nivel, el ZASM proporciona el lenguaje ensamblador Z80 para el desarrollo de programas en lenguaje máquina.

Como ayuda al desarrollo de la habilidad de teclado y proceso de textos se proporcionan cuatro programas diferentes. El Touch'n'Go está diseñado para desarrollar la destreza en la mecanografía al tacto. WORD es un curso de proceso de textos para principiantes, concebido para enseñar al alumno los principios y las técnicas que se emplean en el tratamiento de textos. Para una implementación completa, en el lote también se incluye el WordStar. TXED es un editor de textos que se puede utilizar ya sea para tratamiento de textos o bien para desarrollo de programas.

Quest-D es una base de datos diseñada para enseñar los principios del almacenamiento y la recuperación de datos. Más especializado es el SIR, destinado a catalogar los recursos de la biblioteca de la escuela y para enseñar las técnicas de las funciones de gestión de una biblioteca.

Por último, Telesoftware es un sistema de videotexto particularmente útil al utilizarlo junto con las capacidades de red del 480Z.




Chris Stevens

## LINK 480Z

**DIMENSIONES**

520 × 330 × 80 mm

**CPU**

Z80, operando a 4 MHz

**MEMORIA**

64 K de RAM

**PANTALLA**

Texto: 80 × 25 caracteres
Resolución media: 160 × 192 con 16 colores
Alta resolución: 320 × 192 con cuatro colores
Resolución ultra alta: 640 × 192 con dos colores

**INTERFACES**

Puerta RF, pantalla, RGB/TTL, puerta accesoria, cassette, puerta en paralelo, dos puertas en serie, red, enchufe de video para conexión en red

**LENGUAJES DISPONIBLES**

BASIC, LOGO y PASCAL

**TECLADO**

65 teclas, incluyendo teclas de función y de control del cursor

**DOCUMENTACION**

Los manuales son exhaustivos y contienen toda la información necesaria tanto para el principiante como para el usuario avanzado. Sin embargo resulta difícil hallar parte de la información

**VENTAJAS**

El Link 480Z se ha desarrollado para su uso en la clase y en muchos sentidos es ideal para las escuelas. La capacidad para ejecutar CP/M y la riqueza de software disponible, junto con sus capacidades para conexión en red, le confieren gran versatilidad. El ordenador está bien construido y puede servir durante muchos años

**DESVENTAJAS**

Se trata de un ordenador anticuado que, al compararlo con máquinas que ofrecen capacidades similares, parece tener un precio excesivo

# Paquetes eficientes

## Examinemos cuatro programas destinados a microordenadores personales: "Micro Swift", "Practicalc II", "PS" y "Vizastar"

*Micro Swift, Practicalc II, PS* y *Vizastar* pertenecen a la nueva clase de paquetes perfeccionados basados en hoja electrónica, que evidentemente están inspirados en el paquete integrado *Lotus 1-2-3* y en su sucesor, el *Symphony*. Pero mientras que el *1-2-3* y el *Symphony* se escribieron para el IBM-PC y máquinas compatibles (para ejecutar el *1-2-3* se requieren 296 K de memoria para el usuario, y el *Symphony* exige un mínimo de 320 K), los nuevos paquetes están diseñados para micros personales. En muchos sentidos, los cuatro paquetes que vamos a analizar han hecho milagros para comprimir muchas de las características disponibles en paquetes más grandes en los aproximadamente 30 K de memoria de que dispone el usuario de micros tales como el Commodore 64.

Sin embargo, por el momento estos paquetes "miniSymphony" sólo pueden ofrecer dos de las cuatro opciones que hacen que los paquetes más potentes (y más caros) sean tan atractivos. Dadas las actuales limitaciones de hardware, intentar incorporar las cuatro opciones (hoja electrónica, base de datos, tratamiento de textos y posibilidad de programación) sin duda alguna habría exigido el tipo de concesiones que han hecho que el software basado en ROM Three-Plus-One del Commodore Plus/4 se haya convertido en algo decepcionante.

## Ventajas relativas

Consideremos algunas de las opciones que ofrecen estos cuatro programas con el fin de comparar sus ventajas relativas. El *PS*, el *Micro Swift* y el *Vizastar* son, en mayor o menor grado, programables. Ésta es una facilidad sumamente valiosa, puesto que permite al usuario automatizar funciones cuya ejecución, de lo contrario, requeriría muchas pulsaciones de teclas; ello se consigue de la misma forma en que se emplean las macros de teclado en el *Lotus 1-2-3*.. Los tres programas lo hacen de forma diferente; analizaremos estos enfoques uno por uno.

En el paquete *PS* los módulos se programan empleando instrucciones familiares del BASIC. Estos módulos se guardan luego pulsando <f3> y se ejecutan utilizando <U>, o bien pueden ejecutarse automáticamente después de la carga si son salvados en disco con un punto y aparte después del nombre del programa. El paquete posee toda una gama de útiles facilidades de programación: por ejemplo, puede bifurcar a una subrutina de un programa desde una fórmula dentro de una celda simplemente insertando dentro de la fórmula la instrucción GOSUB. Pueden definirse funciones utilizando la función FN, y el programa también dispone de la facilidad de pasar series de caracteres, fila y columna, y valores numéricos.

El *Micro Swift* se puede programar colocando una lista de instrucciones en la columna Z; la primera instrucción da el nombre del programa, precedido por un signo numérico (#) y la última línea contiene la instrucción @ QUIT. Veamos un ejemplo sencillo:

```
Z1 # SUM              (sumar)
Z2 @ SUM(A1,A3)       (sumar)
Z3 @ ASSIGN(Z2,A4)    (asignar)
Z4 @ QUIT             (salir)
```

Este programa sumará los valores contenidos en las celdas A1, A2 y A3, y después le asignará el resultado, que ahora se halla en la celda Z2, a la celda A4. El programa es llamado mediante la instrucción # SUM.

De todos los paquetes mencionados aquí, quizá el más sencillo sea el *Vizastar*, puesto que las instrucciones consisten en las letras iniciales que se pulsarían para ejecutarlas de forma manual. Por lo tanto, para utilizar una base de datos específica, se pulsaría la tecla CBM seguida de D(ata: datos), U(se: usar), D(atabase: base de datos) y el nombre de la base de datos. Por último, pulsaría <RETURN>. Para programar, se utiliza el signo de barra (/) en lugar de la tecla CBM, de modo que /DUDnombre-[RET] ejecutará la acción si se pulsa <F8>. Las teclas de función y edición se programan pulsando <CRTL> más la tecla apropiada, y esta letra se imprime cuando se utiliza la función en un programa. No obstante, cuando se programan de esta manera las teclas del cursor, éstas se imprimen como [up] (arriba), [down] (abajo), [left] (izquierda) o [right] (derecha).

La base de datos del *Vizastar* es una implementación muy potente que en realidad utiliza una sección de la hoja que no está disponible para el usuario (filas de la 1 000 en adelante) para almacenar formatos de registros. Cada registro puede constar de hasta nueve pantallas, y éstos pueden ser accedidos mediante las instrucciones Key o Next, Prior, First, Last o Current (utilizando cada una la letra inicial de un menú de instrucciones). Asimismo, los registros se pueden Add (sumar), Replace (modificar) o Delete (eliminar).

Los campos tienen nombres de letras, empezando con la A y terminando con BK, que aluden a las columnas de ese nombre en la hoja electrónica. Por consiguiente, a modo de ejemplo, los criterios de búsqueda se pueden preparar en una línea vacía de la hoja electrónica. A siempre es el campo clave, es decir, el campo en el cual se clasifican los datos.

*Practicalc II* es una hoja electrónica que utiliza una facilidad de "etiqueta larga", que permite disponer texto "a caballo" de varias celdas. Esta facilidad permite que el programa opere como un procesador de texto con una longitud de línea máxima de 100 caracteres. Esta opción dispone de la mayoría de las facilidades más comunes de tratamiento de textos, incluyendo desplazamiento de palabras,

# Transporte informatizado

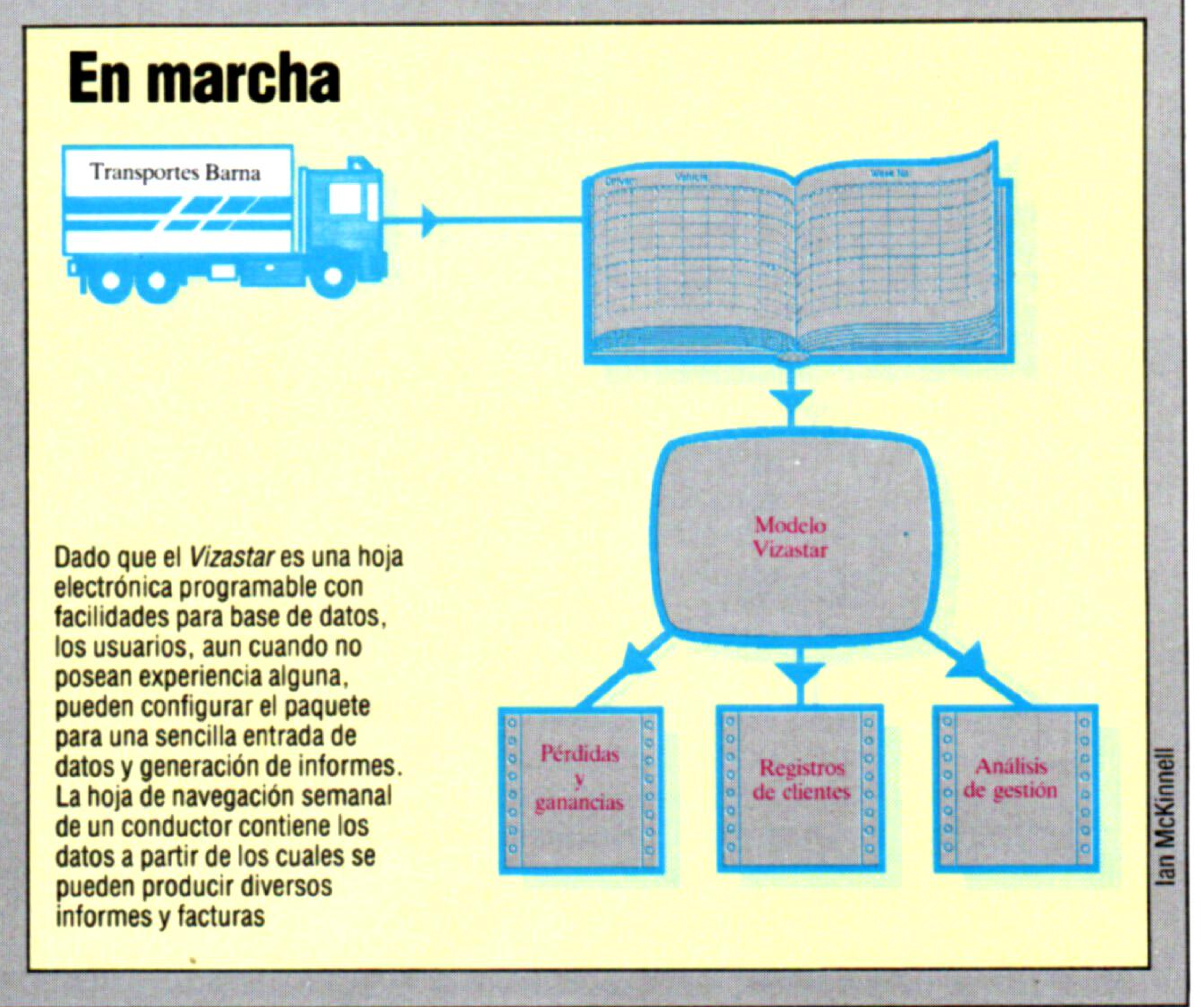
La mayoría de las flotas de camiones en Gran Bretaña están compuestas por unos cinco vehículos. Pero la mayor parte de los paquetes para ordenador disponibles para administrarlas están diseñados para una cantidad mayor de camiones. El paquete para administración de flotas *MEM Computing*, por ejemplo, puede manipular flotas de 1 000 o más vehículos.

No es sorprendente que, por este motivo, sean pocos los propietarios de flotas pequeñas que hayan optado por informatizar sus operaciones, tal como descubrió Terry Palmer, consejero de transportes británico, cuando llevó a cabo un estudio de mercado patrocinado por el Science and Engineering Research Council. A consecuencia del mismo, Palmer se abocó a la tarea de crear un sistema que tuviera un sentido más comercial. Aunque comenzó a desarrollarlo utilizando el *Lotus 1-2-3*, acabó por hacerlo caber en la memoria de un Commodore 64 empleando el *Vizastar*, una combinación de hoja electrónica y base de datos programable. El costo total, según ha calculado Palmer, ha ascendido a 1 000 libras (unas 200 000 ptas), incluyendo software y hardware: alrededor de la quinta parte del costo total de los sistemas más grandes.

La investigación la realizó Palmer como parte de un proyecto que está llevando a cabo en asociación con el Polytechnic de Central London, para ver si a los pequeños camioneros la información que les proporciona tal sistema les resulta útil y si estarían preparados para invertir en la misma. Comenzó con la hoja de navegación que usan todos los transportistas y terminó con formularios de informes en los cuales los camioneros registran todos los trabajos efectuados, sus recorridos, destinos, gastos de viaje, costo de combustible, gastos en efectivo y costos operativos. Al final de cada semana, los datos de las hojas se transfieren a la hoja electrónica.

Al finalizar la entrada de datos, la hoja ya ha calculado si se han obtenido pérdidas o beneficios, y produce un análisis completo de la semana comercial. Las semanas se pueden consolidar posteriormente en análisis mensuales, y los meses en un total anual. Dado que el *Vizastar* también trata una parte de la hoja como si fuera una base de datos, la grabación de registros de disco y su recuperación es exactamente igual que en los programas exclusivos de bases de datos (utilizando un campo de clave o posibilitando la revisión de la lista mediante el empleo de las instrucciones Next, Prior o Current, First o Last), con lo que se puede llevar un registro permanente de clientes. También en la hoja se pueden incluir anotaciones.

## En marcha




Dado que el *Vizastar* es una hoja electrónica programable con facilidades para base de datos, los usuarios, aun cuando no posean experiencia alguna, pueden configurar el paquete para una sencilla entrada de datos y generación de informes. La hoja de navegación semanal de un conductor contiene los datos a partir de los cuales se pueden producir diversos informes y facturas

desplazamiento de bloques, inserción y supresión.

También se puede cargar una hoja electrónica en una parte de un documento. La hoja electrónica seguiría estando "activa" (lo que significa que sus fórmulas, valores u otros contenidos pueden ser modificados en el documento principal, sin, por supuesto, afectar a la hoja en disco).

Si bien las limitaciones de memoria imponen que se pueda acceder sólo a un par de opciones dentro de cualquiera de estos programas, los cuatro pueden acceder a archivos de tratamiento de textos o base de datos producidos mediante otros programas editados por la misma empresa. Por ejemplo, el *Vizastar* puede manipular archivos de tratamiento de textos generados mediante el *Vizawrite*; el *Micro Swift* puede acceder a archivos de base de datos producidos por el *Micro Magpie*, y el *Practicalc* y el *PS* pueden utilizar archivos del *Practifile* de Practicorp. En realidad, puesto que todos utilizan formatos secuenciales, pueden acceder a archivos mutuos, así como a programas absolutamente dispares, como el procesador de textos *Easy script*. Aunque no puede decirse que esto sea exactamente una integración completa de software, sí nos aproxima mucho a ella.

**Micro Swift:** Para el Commodore 64
**Editado por:** Audiogenic, PO Box 88, Reading, Berks., Gran Bretaña
**Formato:** Disco

**Practicalc II:** Para el Apple II de 48 K, el BBC Micro y el Commodore 64
**Editado por:** Practicorp, Goddard Road, Whitehouse Ind Est, Ipswich, Suffolk IP1 5NP, Gran Bretaña
**Formato:** Disco

**PS:** Para el Commodore 64
**Editado por:** Practicorp, Goddard Road, Whitehouse Ind Est, Ipswich, Suffolk IP1 5NP, Gran Bretaña
**Formato:** Disco

**Vizastar:** Para el Commodore 64
**Editado por:** Viza Software, 9 Mansion Row, Brompton, Gillingham, Kent ME7 5SE, Gran Bretaña
**Formato:** Disco con cartucho de 4 K

# Control exacto

## Llegados a este punto, estudiaremos el calibrado del robot para lograr un preciso control en sus movimientos

Los motores paso a paso son ideales para el control mediante dispositivos digitales, dado que giran un intervalo preciso cada vez que reciben un impulso. Para establecer una relación entre el control digital del motor paso a paso y el mundo real de distancias y ángulos, hemos de llevar a cabo algunos experimentos iniciales con nuestro robot. Éstos nos permitirán determinar la cantidad de impulsos necesarios para moverlo a través de varios ángulos y distancias. Luego de realizar estos experimentos estaremos en condiciones de determinar las relaciones promedio de impulso/distancia e impulso/ángulo, que deberemos entrar en los programas a modo de constantes. En futuros capítulos diseñaremos software que, unido a otras aplicaciones, permitirá al robot sondear y construir representaciones digitales de objetos sólidos. Para conseguir que funcione con precisión necesitaremos los valores de las relaciones obtenidas en los experimentos realizados con anterioridad en este apartado.

## Calibrado lineal

Podemos hacer una hipótesis de la relación impulso/distancia de nuestro robot valiéndonos de la matemática elemental. Puesto que un impulso produce en los motores un giro de 7,5°, colocar la salida del motor en un coeficiente de reducción de 25:2 significará que un impulso producirá un giro de 7,5 × 2/25 = 0,6° en el eje. Puesto que la rueda Lego tiene un radio de 30 mm, el movimiento lineal por impulso se puede calcular del siguiente modo: 1 impulso produce un movimiento de 0,6/360 × 2 × *pi* × 30 mm. Descomponiendo esta expresión encontramos que 1 impulso produce un movimiento de 0,1 × *pi* mm. La inversión de esta cifra nos da un coeficiente teórico de impulso/distancia: coeficiente i/d=3,183.

El programa de calibrado que ofrecemos a continuación le permitirá efectuar varias pruebas con su robot a través de diversas distancias. A cada ejecución se visualizan en la pantalla la cantidad de impulsos y las distancias teóricas que se habrán recorrido. Utilizando dos reglas de 30 cm dispuestas una a continuación de otra, se puede registrar la distancia real recorrida en cada prueba. El programa visualiza luego una tabla de la cantidad de impulsos, las distancias reales registradas y los cálculos teóricos. También se calcula un coeficiente i/d promedio. Esta cifra es importante, de modo que conviene apuntarla por separado. La muestra de salida de este programa indica que nuestro robot prototipo tiende a recorrer, para un número dado de impulsos, una distancia ligeramente mayor de lo que sugeriría la teoría. La importancia de este ejercicio radica en que usted halle el coeficiente i/d para su robot y lo utilice en futuros programas.

```
 10 REM **** CALIBRADO BBC ****
 20 RDD=$FE62:REGDAT=&FE60
 30 ?RDD=15:REM LINEAS 0-3 SALIDA
 50 adelante=4:atras=2:DIM MD(12)
 60 FOR CC=500 TO 1700 STEP 100
 70 ?REGDAT=0
 80 ?REGDAT=(?REGDAT OR 1) OR adelante
 90 PRINT CC,INT(CC*PI)/10
100 A$=GET$
110 FOR I=1 TO CC
120 PROCimpulso
130 NEXT I
140 INPUT"DISTANCIA MEDIDA EN MM";MD((CC-500)/100)
150 NEXT CC
160 ?REGDAT=0:T=0
180 PRINT"     IMPULSOS","   MEDIDOS", "   TEORICOS."
190 PRINT
200 FOR CC=500 TO 1700 STEP 100
210 PRINT CC,MD(CC-500)/100),INT(CC*PI)/10
220 T=T+CC/MD((CC-500)/100)
230 NEXT CC
240 PRINT:PRINT " COEFICIENTE IMPULSO/DISTANCIA:",T/12
260 END
270 DEF PROCimpulso
280 ?REGDAT=(?REGDAT OR 8)
290 ?REGDAT=(?REGDAT AND 247)
300 ENDPROC
```

```
 10 REM **** CALIBRADO CBM 64 ****
 20 RDD=56579:REGDAT=56577
 30 POKE RDD,15:REM LINEAS 0-3 SALIDA
 50 AD=4:AT=2:DIM MD(12)
 60 FOR CC=500 TO 1700 STEP 100
 70 POKE REGDAT,0
 80 POKE REGDAT,(PEEK(REGDAT)OR 1)OR AD
 90 PRINT CC,INT(CC*π)/10
100 GET A$:IF A$=" " THEN 100
110 FOR I=1 TO CC
120 GOSUB 270:REM IMPULSO
130 NEXT I
140 INPUT"DISTANCIA MEDIDA EN MM";MD((CC-500)/100)
150 NEXT CC
160 POKE REGDAT,0:T=0
170 REM ** LINEAS 180-260 IGUALES QUE VERSION BBC **
175 REM ** PERO REEMPLAZAR PI POR π EN LA LINEA 210 **
270 REM **** S/R IMPULSO ****
280 POKE REGDAT,PEEK(REGDAT)OR 8
290 POKE REGDAT,PEEK(REGDAT)AND 247
300 RETURN
```

## Calibrado angular

Podemos calcular un coeficiente impulso/ángulo de este modo: si la distancia entre ejes es de 140 mm, la circunferencia del círculo de giro = 140×*pi*. Un impulso produce un giro de 360×0,1×*pi*/(140×*pi*) grados y, por tanto, el coeficiente i/a es 3,846.

Uno de los principales problemas que implica el calibrado angular es la medición exacta de los ángulos. Dado que en la mayor parte de las aplicaciones el robot girará 90° (o múltiplos de 90), nuestro coeficiente teórico de i/a nos indica que se requerirían 346 impulsos para un giro en ángulo recto.

Marque en un trozo de papel un par de líneas perpendiculares. En una de ellas haga dos pequeñas marcas a cada lado del punto donde se intersectan ambas, para designar los puntos de partida de las ruedas del robot. Ejecute el siguiente programa para hacer que el robot gire 90°. El bucle FOR...NEXT de la línea 70 dicta la cantidad de impulsos que se le pasan a los motores. La cifra 371 es el valor experimental requerido para que nuestro robot prototipo gire 90°. Edite el programa, alterando el límite superior de este bucle, hasta que las ruedas de su robot queden alineadas exactamente con la otra línea perpendicular dibujada sobre el



OBSTÁCULO

CONTADOR

ADELANTE

Impulso

Cont.=cont.+1

Sensor=?(Regdat)AND 192

Es el sensor =192?

STOP

ATRAS

For K=cont. TO cont.+1

Impulso

Next K

STOP

PUNTO DE PARTIDA

Kevin Jones

| IMPULSOS | MEDIDOS | TEÓRICOS |
|---|---|---|
| 500 | 160 | 157 |
| 600 | 195 | 188,4 |
| 700 | 225 | 219,9 |
| 800 | 258 | 251,3 |
| 900 | 293 | 282,7 |
| 1000 | 324 | 214,1 |
| 1100 | 352 | 345,5 |
| 1200 | 390 | 376,9 |
| 1300 | 421 | 408,4 |
| 1400 | 452 | 439,8 |
| 1500 | 488 | 471,2 |
| 1600 | 522 | 502,6 |
| 1700 | 553 | 534 |

COEFICIENTE IMPULSOS DISTANCIA: 3,34767511

**Tabla teórica**
El programa de calibrado del robot produce esta tabla, que se debe verificar para asegurar que cada distancia medida guarde una relación razonable con la distancia teórica correspondiente. Se calcula un coeficiente i/d global como el promedio de los coeficientes de i/d obtenidos en las 12 pruebas. Este número se debe conservar, porque será necesario en futuros programas

## Adelante y atrás

El movimiento hacia adelante se efectúa estableciendo los bits de dirección hacia adelante de los activadores del motor paso a paso y enviándoles un impulso a los motores. Se va incrementando un contador de impulsos y se comprueban los bits de sensores del registro de datos en busca de una entrada de los sensores de colisión. Este proceso se repite hasta que se detecta una colisión, momento en el cual se invierten los bits de dirección del motor y un bucle FOR...NEXT envía los impulsos necesarios para devolver al robot hasta su punto de partida. El movimiento del robot hacia adelante es más lento y ligeramente más torpe que el de regreso

papel. Se pueden efectuar otras comprobaciones. Reemplace la dirección "derecha" (DE) por "izquierda" (IZ) en la línea 60 y asegúrese de que el robot también gira 90º en sentido antihorario. De duplicar el valor superior del bucle FOR...NEXT, se produciría un giro de 180º. Compruebe que las ruedas terminan en los mismos puntos en los cuales empezaron. De no ser así, es necesario un ligero ajuste de las ruedas para asegurar que estén situadas simétricamente a cada lado del eje central. Cuando esté satisfecho con la posición de las ruedas, marque sus posiciones en el eje y péguelas en su sitio.

```
10 REM **** GIRO CBM 64 ****
20 RDD=56579:REGDAT=56577
30 POKE RDD,15:REM LINEAS 0-3 SALIDA
40 IZ=6:DE=0
50 POKE REGDAT,0
60 POKE REGDAT,(PEEK(REGDAT)OR 1)OR DE
70 FOR I=1 TO 371:GOSUB 90:NEXT I
80 POKE REGDAT,0:END
90 REM **** S/R IMPULSO ****
100 POKE REGDAT,PEEK(REGDAT)OR 8
110 POKE REGDAT,PEEK(REGDAT)AND 247
120 RETURN
```

```
10 REM **** GIRO BBC ****
20 RDD=&FE62:REGDAT=&FE60
30 ?RDD=15:REM LINEAS 0-3 SALIDA
40 izquierda=6:derecha=0
50 ?REGDAT=0
60 ?REGDAT=(?REGDAT OR 1)OR derecha
70 FOR I=1 TO 371:PROCimpulso:NEXT
80 ?REGDAT=0:END
90 DEF PROCimpulso
100 ?REGDAT=(?REGDAT OR 8)
110 ?REGDAT=(?REGDAT AND 247)
120 ENDPROC
```

Ahora que hemos incorporado a nuestro robot sensores de microinterruptor podemos escribir software que utilice la salida de la puerta para el usuario para controlar el robot, y la entrada para monitorizar las actividades externas a través de los sensores del robot. El siguiente y sencillo programa envía al robot hacia adelante hasta encontrar un objeto, momento en el cual el robot retrocede exactamente hasta su posición de partida. La lógica del programa se puede describir del siguiente modo:

1. Establecer el registro de dirección de datos en 15. Con ello se establecen los bits 0–3 como salida y los bits 4-7 como entrada.

2. Establecer la dirección del robot hacia adelante.
3. Impulsar los motores hasta poner bajo el bit 6 o el bit 7, llevando la cuenta de la cantidad de impulsos efectuados.
4. Establecer la dirección del robot hacia atrás.
5. Impulsar los motores las veces que indique el "contador".
6. Establecer el reg. de datos en 0 y terminar.

```
 10 REM **** PARACHOQUES CBM ****
 20 RDD=56579:REGDAT=56577
 30 POKE RDD,15:REM LINEAS 0-3 SALIDA
 40 AD=4:AT=2
 50 POKE REGDAT,(PEEK(REGDAT)OR 1)OR AD
 60 REM **** IMPULSO ADELANTE ****
 65 CC=0
 70 GOSUB 1000:CC=CC+1:REM IMPULSO
 80 IF (PEEK(REGDAT)AND 192)=192 THEN 70
 90 REM **** REGRESAR A PARTIDA ****
 95 POKE REGDAT,(PEEK(REGDAT)AND 1)OR AT
100 FOR I=1 TO CC
110 GOSUB 1000:REM IMPULSO
120 NEXT I
130 POKE REGDAT,0:END
1000 REM **** S/R IMPULSO ****
1010 POKE REGDAT,(PEEK(REGDAT)OR 8)
1020 POKE REGDAT,(PEEK(REGDAT)AND 247)
1030 RETURN
```

```
 10 REM **** PARACHOQUES BBC ****
 20 RDD&FE62:REGDAT=&FE60
 30 ?RDD=15:REM LINEAS 0-3 SALIDA
 40 adelante=4:atras=2
 50 ?REGDAT=(?REGDAT OR 1) OR adelante
 60 REM **** IMPULSO ADELANTE ****
 65 contador=0
 70 REPEAT:PROCimpulso:contador=contador+1
 80 UNTIL (?REGDAT AND 192)<>192
 90 REM **** REGRESAR A PARTIDA ****
 95 ?REGDAT=(?REGDAT AND 1)OR atrás
100 FOR I=1 TO contador
110 PROCimpulso
120 NEXT I
130 ?REGDAT=0:END
1000 DEF PROCimpulso
1010 ?REGDAT=(?REGDAT OR 8)
1020 ?REGDAT=(?REGDAT AND 247)
1030 ENDPROC
```

## Ajustes preliminares

Antes de comenzar el proceso de calibrado es necesario efectuar algunos ajustes. Primero, con el robot patas arriba, es importante situar correctamente el eje central sobre el cual girará éste. Esto se puede hacer mediante las mediciones que se indican. Marque el centro del eje con un pequeño rasguño o con un rotulador indeleble. Mida la distancia entre las caras interiores de las ruedas (debe estar entre 140 y 150 mm). Es importante, para que el robot pivote correctamente, que cada rueda sea equidistante respecto al punto central del eje que hemos marcado. Deslice suavemente las ruedas a lo largo de sus ejes hasta lograrlo



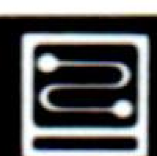
Kevin Jones

En este programa hemos designado al par de interruptores hacia adelante como el par que está más alejado de los conectores de parche de la tapa del robot, y hemos conectado estos dos microinterruptores a los bits 6 y 7, utilizando dos cables entre los dos pares de conectores situados más a la derecha de la tapa, el rojo y el azul. En el futuro daremos siempre por sentado que el enchufe D está aún más adelante que el sistema de conectores de parche. Si al ejecutar este programa se encuentra con que su robot avanza primero hacia atrás (de acuerdo a esta convención), quite la tapa y déle la vuelta.

De los cuatro bits bajos del registro de datos que controlan la operación del motor, el bit 0 es el bit reposo (establecido normalmente en uno), los bits 2 y 3 son los controladores de dirección para los motores derecho e izquierdo, y el bit 3 impulsa simultáneamente ambos motores, haciéndolos girar un paso cada vez que el bit 3 sufre una transición de bajo a alto. El empleo de los operadores lógicos AND y OR permite encender y apagar bits individuales sin afectar a los otros bits del registro. Dado que los cuatro bits superiores han sido establecidos como entradas por el registro de dirección de datos, se suelen mantener altos. Cuando un microinterruptor se cierra, el bit correspondiente del registro de datos pasa a bajo. Normalmente, los bits 6 y 7, si estuvieran establecidos para entrada, tendrían el valor 192 (128+64). El bucle repetitivo que envía el robot hacia adelante en las líneas 70-80, termina con la condición de que estos dos bits ya no posean un valor de 192. Esto puede suceder si se cierra un microinterruptor (o ambos). Si se lleva un contador de la cantidad de impulsos enviados a los motores en el período intermedio, entonces el robot puede retroceder exactamente hasta su punto de partida mediante la alteración de los bits de dirección del motor e impulsando a los motores la cantidad apropiada de veces. El paso de 7,5° de los motores se traduce en un movimiento de menos de 1 mm para la rueda; en consecuencia, podemos controlar muy fácilmente la posición del robot.

Por último, es interesante destacar que el robot se mueve hacia adelante más despacio que cuando se mueve hacia atrás. Aquí nos vemos limitados por la velocidad del BASIC. El tiempo entre impulsos en el bucle que envía al robot hacia adelante es mayor que el que lo hace retroceder, porque en el primer bucle se ha de llevar a cabo un trabajo adicional, como llevar el contador y comprobar la colisión, lo que no sucede en el segundo bucle.

Ahora haremos un alto en el proyecto del robot para darle tiempo para completar el ensamblaje.



# Entreacto

**Finalizamos nuestra introducción al sistema operativo del BBC concentrándonos en la utilización de vectores y examinando la interacción con el ordenador mediante el teclado y la unidad de representación visual (VDU)**

Se dice que la mayoría de las rutinas del OS del BBC están *vectorizadas*. El OS, cuando se le instruye para que llame a la rutina OSCLI, lo primero que hace es llamar a una rutina en la dirección $FFF7. Es ésta la rutina que llama a su vez a la OSCLI, pero no directamente sino que halla la dirección donde se encuentra la OSCLI por medio del contenido de dos bytes de memoria que se encuentran en la página 2 de la RAM. Estos dos bytes son llamados *vector*: el byte inferior de la dirección en cuestión se encuentra en el byte de inferior numeración de dicho vector, y el byte superior de la dirección se halla en el byte de numeración superior. Así, para OSCLI, que se vectoriza mediante las posiciones $208 y $209, el byte inferior de su dirección se halla en la posición $208 y el superior en la $209. Ésta es la conocida convención de direccionamiento *lo-hi* que siguen para su almacenamiento en memoria todas las máquinas 6502. Las direcciones contenidas en cada vector son establecidas por el OS cuando se reinicializa la máquina. ¿Cuál es la justificación de esta alambicada manera de llamar a una rutina del OS? Desde luego no reside en que Acorn esté decidida a hacer la vida del programador lo más complicada posible. Todo lo contrario, el proceso está pensado para facilitársela. ¿Es esto verdad?

Habrá notado que todas las rutinas del OS del BBC mencionadas hasta ahora son llamadas a una dirección contenida en el intervalo $FF00 a $FFFF. No es casual. Una vez llamada esta dirección, se entra en una rutina que ocasiona el salto a la dirección contenida en el vector para esa determinada rutina del OS, como tuvimos ocasión de ver con las llamadas a CLI y a OSCLI. Ahora bien, la dirección que llamamos entre $FF00 y $FFFF es la misma en *todas* las versiones del OS del BBC y así continuará siendo. Si es necesario cambiar los programas internos de la ROM de OS, los diseñadores del OS no tienen más que asegurarse de que las direcciones de las rutinas ROM colocadas en las posiciones del vector han sido alteradas teniendo en cuenta el cambio. Así queda protegido el usuario de tales cambios en el OS siempre que las rutinas del OS sean llamadas en sus puntos correctos de entrada. El contenido, pues, de un vector puede diferir de una versión a otra, pero el usuario no tendrá noticia de ello siempre que use las direcciones del punto de entrada en el intervalo $FF00 a $FFF.

Una segunda ventaja del empleo de vectores es que el método nos brinda un medio de modificar el comportamiento de las rutinas del OS. Basta con alterar el contenido de un vector de modo que apunte a una rutina en código máquina diseñada por nosotros mismos, interceptando así las llamadas normales del OS. Más adelante examinaremos los vectores empleados en las principales rutinas del sistema operativo.

De momento, consideraremos el vector llamado USERV, al que apuntan las posiciones $200 y $201. Se trata de un vector bastante especial, ya que por lo general no hace nada. Se emplea con dos órdenes *, las *CODE y *LINE. Digítelas y se encontrará con el mensaje de error Bad Command. Pero antes de enviar una carta de queja por este nuevo error del OS del BBC lea lo que sigue.

USERV nos permite definir la función realizada con las instrucciones *CODE y *LINE, o sea, instrucciones definidas por el usuario. *CODE es una manera útil de pasar parámetros a programas en código máquina, como se indica en la tabla siguiente:

| Registro | *CODE x,y | *LINE cadena texto |
|---|---|---|
| A | 0 | 1 |
| X | Contiene el valor del primer parámetro después de *CODE. El parámetro debe estar entre 0 y 255 | Contiene el byte inferior de la dir. de memoria donde se puede hallar el primer car. de la cadena después de *LINE |
| Y | Contiene el valor del segundo parámetro después de *CODE. Otra vez, debe estar entre 0 y 255 | Contiene el byte superior de la dir. de memoria donde se encuentra el primer car. de la cadena después de *LINE |

Esta tabla muestra el estado de los tres registros de la CPU al entrar en la rutina a la que apunta el contenido de USERV. Se tiene un 0 o bien un 1 en A, para indicar cuál de las dos instrucciones provocó la entrada a USERV. Los valores contenidos en X e Y dependen de la instrucción usada, *CODE o *LINE. Por ejemplo, *CODE 3,2 entrará a la rutina que indique USERV con un 0 en A, un 3 en X y un 2 en Y. Es claro que la rutina señalada por la dirección contenida en USERV será la rutina a la que deseamos pasar los dos parámetros.

El programa que proporcionamos en la página siguiente ilustra un sencillo uso de la instrucción *CODE. La rutina en código máquina se ensambla en la memoria empezando en la dirección $C00 como resultado de la sentencia de asignación de la línea 40: la variable entera P% se asocia al contador de programa del procesador, al igual que hacen A% con A, X% con X e Y% con Y, los registros del procesador. Se activa USERV para que apunte a la rutina colocando el byte inferior de esta dirección, $00, en

**Instrucciones definidas por el usuario (USERV)**

```
5   DIM C 100
10  USERV=&200
20  ?USERV=&00
25  ?(USERV+1)=&0C
30  FOR I%=0 TO 2
    STEP 2
40  P%=&C00
50  [ OPT I%
60      CMP #0
70      BNE notcode
80      TXA
90     .loop
95      JSR &FFE3
100     DEY
110     CPY #0
120     BNE loop
130     RTS
140    .notcode
145     RTS
150 ]: NEXT I%
160 .
200 FOR rep=1 to 10
210 FOR asc=33 TO 48
220 asc$=STR$(asc)
230 rep$=STR$(rep)
240 code$="*CODE
    "+asc$+","+rep$
250 $C=code$
260 X%=C MOD 256
270 Y%=C DIU 256
280 CALL &FFF7
290 NEXT: NEXT
```

la posición $200, y el byte superior, $0C, en la $201. La rutina visualiza en la pantalla un determinado número de caracteres; el primer parámetro de la instrucción *CODE contiene el código ASCII del carácter y el segundo parámetro corresponde al número de veces que el carácter será visualizado en la pantalla.

*LINE no es de tan amplia utilidad como *CODE, pero si se desea emplearla se pueden aplicar los principios ilustrados en el siguiente programa, siempre que no se olvide entrar al programa con un 1 en el registro A y que los registros X e Y han de apuntar a la cadena de texto en memoria. Ésa es la función principal de *LINE: pasar cadenas de texto a los programas en código máquina. En aquellos casos en que no hay muchos parámetros que pasar a la rutina, estas dos llamadas son el modo más elegante de hacerlo.

La línea 20 del programa activa USERV para que apunte a nuestra rutina en código máquina. El bucle de la línea 90 a la 120 imprime Y veces el carácter cuyo código ASCII contiene el registro A. Si se entra a la rutina por medio de la instrucción *LINE, las líneas 60 y 70 se encargan de detectar el hecho y de abandonar la rutina. Las líneas de la 200 a la 250 generan la instrucción *CODE con los parámetros variables.

# Interacción del usuario

Los principales métodos de diálogo con el microordenador son el teclado y la pantalla de televisión o VDU (*Visual Display Unit*: unidad de representación visual). Vamos a seguir investigando en detalle cómo el sistema operativo del BBC nos permite la interacción con estas dos áreas fundamentales del ordenador.

Comencemos examinando la llamada OS que nos permite leer caracteres desde la corriente de entrada seleccionada en un momento dado. Esta rutina, denominada OSRDCH (*OS read character*: lectura de caracteres), es llamada en la dirección $FFE0 y se vectoriza a través de las posiciones $210 y $211. Dado que acepta caracteres uno a uno desde la corriente de entrada seleccionada, vamos a ver cómo se selecciona dicha corriente. Existen dos maneras principales: el teclado y la entrada RS423. Se seleccionan por medio de una llamada OSBYTE o bien *FX. El cuadro siguiente resume esta instrucción en código máquina y en BASIC:

| Selección de la corriente de entrada | | | | |
|---|---|---|---|---|
| **n** | **Teclado** | **RS423** | **Ensamblador** | **BASIC** |
| 0 | ✔ | ✘ | LDA # 2 | *FX2,n |
| 1 | ✘ | ✔ | LDX # n | |
| 2 | ✔ | ✔ | JSR $FFF4 | |

Por tanto, *FX2,1 desconecta el teclado y conecta la RS435 como corriente de entrada. Los datos recibidos en la RS423 serán tratados como si hubieran sido digitados. En ensamblador, n debe tener valor 1 si se desea obtener el mismo efecto.

Una vez seleccionada la entrada, se puede acceder a esa corriente por medio de OSRDCH. Lo primero que hay que decir sobre esta rutina es que su utilidad real se demuestra sólo en programas en ensamblador, puesto que el BASIC ya tiene rutinas similares con GET e INPUT. Esta rutina debe ser llamada en la dirección $FFE0 y, a la vuelta de la llamada, el carácter leído de la corriente estará contenido en el registro A; si se detectara cualquier error durante la lectura, el flag de arrastre se pondrá a 1, en caso contrario vendrá a 0. Así, C=1 a la vuelta de la rutina OSRDCH indica que el carácter contenido en A no es válido por cualquier motivo. Cuando la lectura es desde el teclado, el error suele ser debido a la pulsación de la tecla Escape. Esto hace que C se ponga a 1 y el valor contenido en A sea 27 (Escape en ASCII). Si usted se da cuenta de la situación deberá actuar en conformidad y con todo cuidado; el OS del BBC está esperando que el programa le indique que ha recibido este error Escape.

Esto lo haremos empleando una llamada a OSBYTE con A=126, cuyo efecto será el de que se limpien varias partes de la zona de trabajo del OS. Esta operación de respuesta al Escape se hace habitualmente de forma automática por el intérprete BASIC si se pulsa la tecla Escape durante una operación de entrada. La rutina que sigue, muy sencilla, lee la corriente actual de entrada y actúa dependiendo de una posible detección de error de Escape.

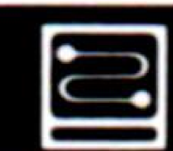
```
1000  .input  JSR  &FFE0
1010          BCS error
1020          RTS
1030  .error  CMP#27
1040          BNE out
1050          LDA#126
1060          JSR&FFF4
1070  .out    RTS
```

La línea 1000 llama a la rutina OSRDCH y la 1010 comprueba el estado del flag de arrastre. Si está a cero, entonces se ejecuta una RTS al programa de llamada, con un carácter válido en el registro A. Si no es así, la línea 1030 comprueba si el error fue debido a Escape, y, en caso afirmativo, las líneas 1050 y 1060 ejecutan la llamada a OSBYTE que se encargará de acusar la recepción del evento Escape. Puede que usted piense que para entrar cadenas de datos en sus programas en código máquina deberá emplear una rutina de fabricación propia, pero no es así. Existe en el OS un medio de leer cadenas de caracteres desde la corriente actual de entrada. A esta rutina se accede por medio de una de las llamadas a OSWORD, que habremos de examinar más adelante en este curso. No obstante, usaremos ahora esta llamada particular de OSWORD para leer cadenas de caracteres.

# El bloque de control

Las rutinas OSWORD son llamadas en la dirección $FFF1. Existen varias, y para especificar cuál necesitamos nos servimos del valor contenido en A en el momento de hacer la llamada. En todas las llamadas a OSWORD, los registros X e Y del 6502 apuntan a un bloque de memoria llamado *bloque de control*, que contiene los parámetros que se pasan a la rutina. El registro X contiene el byte inferior de la dirección del bloque de control y el registro Y contiene el superior, según la norma seguida por el 6502 del *lo-hi*.

La manera en que se inicializa el bloque de control es la siguiente:



| El bloque de control de OSWORD | |
|---|---|
| **Entrada al bloque control** | **Función** |
| 0 | Byte inferior de la dirección en la que van a ser escritos los caracteres |
| 1 | Byte superior de la dirección en la que van a ser escritos los caracteres |
| 2 | Longitud máxima de la línea |
| 3 | Código ASCII mínimo aceptable |
| 4 | Código ASCII máximo aceptable |

Durante la introducción de los caracteres por medio de esta rutina, la tecla Delete tiene su función habitual. La rutina puede ser cancelada con sólo pulsar Return o Escape. Por ejemplo, el bloque de control que sigue a continuación tiene estos efectos cuando se hace una llamada a OSWORD:

| Ejemplo de bloque de control de OSWORD | |
|---|---|
| **Entrada al bloque de control** | **Valor** |
| 0 | &00 |
| 1 | &0C |
| 2 | &07 |
| 3 | 32 |
| 4 | 96 |

1. El primer carácter entrado será almacenado en $C00, el segundo en $C01, etc.
2. Solamente se aceptarán siete caracteres; si intenta digitar algún carácter más, entonces se generará un pitido y todo lo que escriba será ignorado.
3. Solamente se aceptarán códigos ASCII que vayan desde el 32 al 96 (desde el carácter Espacio al carácter £); los demás serán ignorados.

Como puede ver, la llamada nos permite descartar caracteres no deseados a la hora de su entrada. Cuando se ha producido la salida de la rutina, el estado del flag C nos informa de la causa del final de rutina. Si C=1, es que se pulsó Escape. Si C=0, es que Return se encargó de finalizar la entrada de caracteres y el registro Y retiene la longitud de la cadena entrada, incluyendo el valor de retorno de carro en ASCII como final de la cadena. Recuerde que puede servirse de esta rutina en cualquiera de las corrientes de entrada seleccionadas por medio de *FX2 o su equivalente en código máquina.

Acabamos de ver lo fácil que es una lectura de datos en el BBC Micro. Avancemos un poco más examinando los medios de enviar los caracteres a la corriente de salida seleccionada. De nuevo habremos de emplear una llamada del OS para seleccionar la corriente de salida que deseamos usar. Se consigue con *FX3,n, donce n especifica la corriente que se selecciona. Cada bit del parámetro n controla una corriente diferente. Por ejemplo, *FX3,1 permite las salidas seriales, de pantalla o de impresora y permite una salida SPOOL, siempre que haya sido generada una instrucción *SPOOL.

La principal rutina empleada para enviar caracteres a la corriente seleccionada de salida se llama OSWRCH (WR*ite* CH*aracter*: escritura) y es llamada en la dirección $FFEE, vectorizada por medio de las posiciones $20E y $20F. Su empleo es sencillo; basta con cargar el registro A con el código ASCII del carácter que se desea escribir y llamar después la rutina. Las siguientes tres rutinas imprimen el carácter A en pantalla:

| Tabla de parámetros de control corriente salida | | | | | | |
|---|---|---|---|---|---|---|
| Valor \ Bit | 0 | 1 | 2 | 3 | 4 | 6 |
| 1 | RS423 ON | Pantalla OFF | Impresora OFF | Impresora ON* | Spool OFF | Impresora OFF** |
| 0 | OFF | ON | ON | OFF* | ON | ON** |

* Impresora en ON u OFF, esté o no desactivada por otra causa

** Impresora en OFF a menos que el carácter no sea precedido por CHR$(1)

```
1000 VDU 65
1000 PRINT CHR$(65)
1000 LDA#65
1010 JSR &FFEE
```

La orden VDU del BASIC produce en esencia los mismos efectos que OSWRCH. Los caracteres del intervalo ASCII 32 a 255 se visualizan en pantalla, salvo el 127, que es el carácter Delete. Los incluidos en el intervalo del 0 al 31 tienen funciones especiales. Ellos son los que nos permiten emplear OSWRCH para dibujar gráficos en pantalla, ejecutar las instrucciones COLOUR y GCOL, definir caracteres y controlar el chip 6845, que es el que controla la visualización video del BBC Micro.

La escritura de caracteres en pantalla o sobre otro soporte cualquiera a través de la rutina OSWRCH es conocida a menudo como escritura sobre *drivers* (manejadores) de VDU. La tabla de códigos de control ASCII ilustra los efectos de los códigos de carácter entre el 0 y el 31 cuando son enviados a *drivers* o unidades VDU. Notará que éstos nos permiten hacer, por medio de rutinas de gráficos en código máquina, todo lo que podemos hacer en BASIC.

## Gráficos vía la OSWRCH

Las rutinas OSWRCH nos ofrecen toda la posibilidad de gráficos que obtenemos con instrucciones para gráficos. El programa que servirá de segundo ejemplo, escrito al margen de la página siguiente, nos dibujará en pantalla una línea roja. Las líneas de la 50 a la 75 ejecutan una instrucción GCOL 0,1 estableciendo el color rojo de la línea. Las líneas de la 90 a la 150 ejecutan una instrucción PLOT 5,100,100 que equivale a la orden DRAW 100,100. La línea 100 envía el tipo PLOT (el 5, en este caso) a los *drivers* de VDU, seguido de una abscisa de dos bytes, con el byte inferior primero, y una ordenada de dos bytes, igualmente con el byte inferior primero. Se puede ejecutar una instrucción MOVE sustituyendo el 5 por el 4 (pues MOVE no es más que una instrucción PLOT 4,x,y). Otras operaciones gráficas (como la instrucción PLOT 85 para dibujar triángulos) se pueden obtener por los mismos medios. Pero es im-

**Gráficos vía OSWRCH**

```
10 MODE 1
20 FOR I%=0TO2STEP2
30 P%=&C00
40 [OPT I%
50 LDA #18
55 JSR &FFEE
60 LDA #0
65 JSR &FFEE
70 LDA #1
75 JSR &FFEE
80
90 LDA #25
95 JSR &FFEE
100 LDA #5
105 JSR &FFEE
110 LDA #0
115 JSR &FFEE
120 LDA #100
125 JSR &FFEE
130 LDA #0
135 JSR &FFEE
140 LDA #100
145 JSR &FFEE
150 RTS
160 ]:NEXT I%
170 CALL &C00
```

portante recordar que, en el envío de instrucciones PLOT a los *drivers* VDU, éstos esperan recibir cinco bytes después de ser enviado el valor 25 como primer byte; si estos bytes no llegan a recibirse, pueden ocurrir efectos extraños. Esto es válido para toda operación sobre *drivers* de VDU que exija el envío de más de un byte.

El VDU23 es otra instrucción útil. Se emplea para definir caracteres generados por el usuario. Por ejemplo, VDU23,224,255,255,255,255,255,255,255,255 definirá el carácter 224 (por lo general no definido) como un bloque compacto. Los caracteres comprendidos entre los números 244 y 255 en los modos de 0 a 6 pueden redefinirse por el usuario con esta instrucción. De hecho, la instrucción VDU23 en conjunción con una de las llamadas OSBYTE permite al usuario redefinir otros caracteres en el juego de caracteres.

Toda llamada VDU23 no reconocida por el OS (p. ej., VDU23,0...) es pasada por un vector especial situado en $226 y $227. Si se cambia la dirección contenida en este vector, pueden añadirse rutinas propias con la instrucción VDU23.

Un uso más avanzado de la instrucción VDU23 es el de permitir al programador acceder al chip de control de video 6845. Las instrucciones VDU23 toman la forma:

```
VDU23,0,registro,valor,0,0,0,0,0,0
```

aquí registro es el registro 6845 al que deseamos escribir, y valor es el valor que se escribirá en el 6845. Un ejemplo del uso de VDU23 en este sentido es la alteración de dos registros del 6845 en el programa que sigue: dicen al chip cuál es el área de la memoria del ordenador que ha de servir de memoria de video. El programa cambia el inicio de la RAM de video por la dirección $0000. Lo cual muestra la zona de trabajo del OS del BBC sobre la pantalla y pueden verse efectos interesantes. Trate de añadir unas cuantas líneas de código, de dimensionar algunas matrices, etc. La rutina está escrita en BASIC pero se puede convertir fácilmente en ensamblador:

```
10 MODE 0
20 VDU23,0,12,0,0,0,0,0,0,0
30 VDU23,0,13,0,0,0,0,0,0,0
40 VDU28,0,10,30,0:REM establece ventana texto
50 CLS
```

Hay otras dos llamadas del OS relacionadas con OSWRCH. A saber: OSNEWL y OSASCII. La primera, si es llamada en $FFE7, escribe un salto de línea y un retorno de carro en pantalla. OSASCII, llamada en $FFE3, es una variante en OSWRCH, y resulta útil en tratamiento de textos. Cuando se escribe un carácter 13 por medio de esta llamada, el carácter 10 o salto de línea se escribe en la pantalla de salida. No *deberá*, por tanto, ser usada cuando se están escribiendo instrucciones de gráficos a los *drivers* de VDU, ya que puede generarse un CHR$(10) adicional y provocar confusiones.

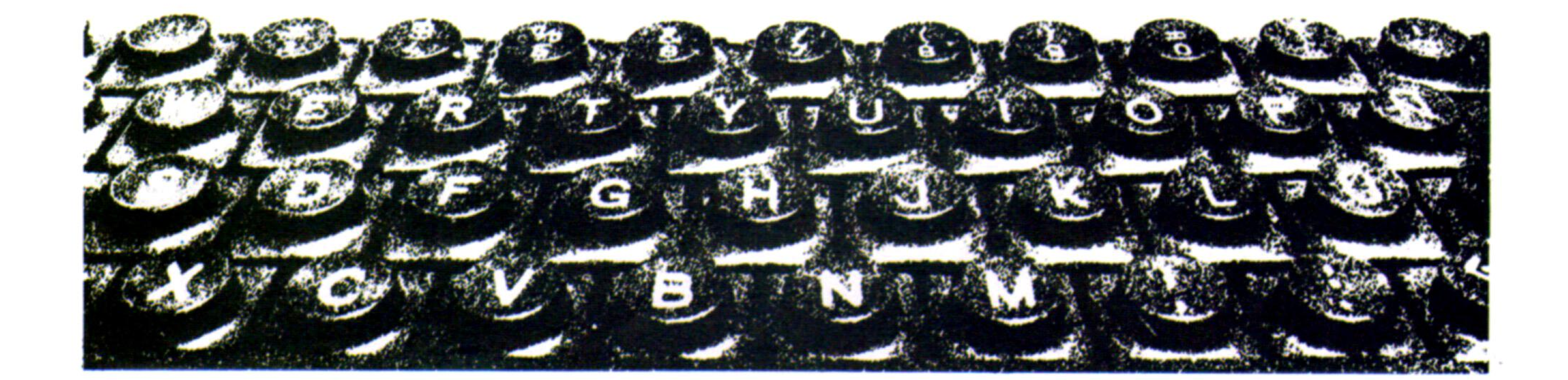
Finalmente *SPOOL y *EXEC son dos instrucciones que permiten salidas y entradas al sistema de archivos seleccionado. *EXEC nombrearchivo abrirá el archivo cuyo nombre se especifique, si existe, leyendo su contenido como si se tratara de un teclado. *SPOOL escribe caracteres al archivo que se cita en la instrucción, como si tales caracteres fueran escritos a la corriente de salida.

Con lo anterior damos por concluida la introducción al sistema operativo del BBC Micro. En los capítulos venideros nos detendremos en el estudio de las rutinas que mejoran la salida en pantalla del Commodore 64, antes de adentrarnos en una amplia exposición de diversos sistemas operativos.

## Tabla cód. de control ASCII

| Código | Bytes adicionales requeridos | Descripción |
|---|---|---|
| 0 | 0 | No hace nada |
| 1 | 1 | Envía el primer carácter a la impresora sólo |
| 2 | 0 | Activa la impresora |
| 3 | 0 | Desactiva la impresora |
| 4 | 0 | Escribe texto al cursor de texto |
| 5 | 0 | Escribe texto al cursor gráfico |
| 6 | 0 | Permite que los drivers de VDU escriban caracteres a la corriente de salida |
| 7 | 0 | Genera un breve sonido |
| 8 | 0 | Mueve el cursor un espacio a la izquierda |
| 9 | 0 | Mueve el cursor un espacio a la derecha |
| 10 | 0 | Mueve el cursor un espacio hacia abajo |
| 11 | 0 | Mueve el cursor un espacio hacia arriba |
| 12 | 0 | Limpia el área de texto de la pantalla |
| 13 | 0 | Devuelve el cursor al inicio de la línea actual |
| 14 | 0 | Activa el modo paginado |
| 15 | 0 | Desactiva el modo paginado |
| 16 | 0 | Limpia el área de gráficos de la pantalla |
| 17 | 1 | Establece el color de texto en el color cuyo código es el byte siguiente |
| 18 | 2 | Hace un GCOL con los dos bytes siguientes que se han de enviar a los drivers. Así, el envío de 18,0,3 ejecutará la orden GCOL 0,3 |
| 19 | 5 | Define los colores lógicos. Ver manual |
| 20 | 0 | Devuelve los colores lógicos a los valores por defecto |
| 21 | 0 | No permite que los drivers de VDU escriban caracteres a la corriente de salida |
| 22 | 1 | Pone el modo de pantalla en el modo del byte siguiente. El envío de 22 y 7 dará Modo 7. La HIMEM no se altera |
| 23 | 9 | Envía instrucciones al chip 6845; programa caracteres definidos por el usuario |
| 24 | 8 | Define una ventana de gráficos |
| 25 | 5 | Realiza la instrucción PLOT |
| 26 | 0 | Establece una ventana de texto y gráficos por defecto |
| 27 | 0 | Ningún efecto |
| 28 | 4 | Establece una ventana de texto |
| 29 | 4 | Establece el origen de gráficos |
| 30 | 0 | Lleva el cursor de texto al ángulo superior izquierdo |
| 31 | 2 | Pone el cursor de texto en la posición *x,y* en los dos bytes siguientes. Así, el envío de 31,10,10 pondrá el cursor en la posición 10,10 de la pantalla |

# Editorial Delta, S.A.



10069

9 788485 822836